ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΚΑΤΑΝΕΜΗΜΕΝΗΣ ΓΝΩΣΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

# ΜΕΛΕΤΗ, ΕΞΑΓΩΓΗ ΚΑΙ ΑΞΙΟΠΟΙΗΣΗ ΣΧΕΣΙΑΚΩΝ ΜΟΝΤΕΛΩΝ ΜΕΤΑΞΥ ΕΙΚΟΝΙΚΩΝ ΟΝΤΟΤΗΤΩΝ ΣΤΟ ΚΟΙΝΩΝΙΚΟ ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σοφία Ελένη Σπαθαριώτη

**Επιβλέπων:** Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

Αθήνα, Ιούνιος 2015

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΚΑΤΑΝΕΜΗΜΕΝΗΣ ΓΝΩΣΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

# **ΜΕΛΕΤΗ, ΕΞΑΓΩΓΗ ΚΑΙ ΑΞΙΟΠΟΙΗΣΗ ΣΧΕΣΙΑΚΩΝ ΜΟΝΤΕΛΩΝ ΜΕΤΑΞΥ ΕΙΚΟΝΙΚΩΝ ΟΝΤΟΤΗΤΩΝ ΣΤΟ ΚΟΙΝΩΝΙΚΟ ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ**

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σοφία Ελένη Σπαθαριώτη

**Επιβλέπων:** Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

| | | |
|---|---|---|
| ..................... | ................ | ...................... |
| Θεοδώρα Βαρβαρίγου | Βασίλειος Λούμος | Συμεών Παπαβασιλείου |
| Καθηγήτρια ΕΜΠ | Καθηγητής ΕΜΠ | Καθηγητής ΕΜΠ |

Αθήνα, Ιούνιος 2015

.......................
Σοφία Ελένη Σπαθαριώτη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

NATIONAL TECHNICAL UNIVERSITY OF ATHENS

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

DIVISION OF COMMUNICATION, ELECTRONIC AND INFORMATION ENGINEERING

DISTRIBUTED KNOWLEDGE AND MEDIA SYSTEMS GROUP

# STUDY, EXTRACTION AND UTILIZATION OF RELATIONAL MODELS AMONG VIRTUAL ENTITIES IN THE SOCIAL INTERNET OF THINGS

THESIS

Sofia Eleni Spatharioti

**Supervisor:** Theodora Varvarigou
Professor, NTUA

Athens, June 2015

## Abstract

The Internet of Things(IoT) is a network of physical objects supplied with unique identifiers and appropriate mechanisms in order to communicate with other objects, without the need of human interaction. This area has grown popular over the years due to a variety of applications, ranging from Home Automation to Healthcare. A recent approach to the IoT is the social Internet of Things(sIoT), where objects communicate based on social characteristics.

The concept that motivated this thesis lies in creating an efficient knowledge flow in the social Internet of Things, through the formation of social relationships. Things acquire information, that is shaped in the form of knowledge through sensors and analysis and are able to learn and expand their knowledge based on other objects' experiences. The ultimate goal of this thesis is to facilitate autonomy in the social Internet of Things by proposing appropriate object relationships to serve the knowledge flow in the network.

After an introduction to the necessary background is provided to the reader, the main subject of this thesis is presented. The first part deals with establishing the main ontologies of the proposed structure of the sIoT; Virtual Entities, Applications and Users. Each ontology is defined by a set of given properties, which will be used in establishing the proposed Relational Models.

Afterwards, the ten proposed Relational Models are analyzed. Each object relationship is explored in detail, by providing information in the form of characteristics, requirements for formation and elements of Relational Models Theory. Motivation and challenges behind each object relationship are also detailed, along with analytic scenarios.

Finally, simulation on a sample network is conducted, through the use of the Gephi platform. Each object relationship is also individually simulated and observations based on simulation results are discussed.

## Keywords

Internet of Things, social Internet of Things, Relational Models, COSMOS, Virtual Entities, knowledge flow, autonomy

## Περίληψη

Το Διαδίκτυο Των Πραγμάτων(ΔτΠ) είναι ένα δίκτυο α π ό φυσικά αντικείμενα εξοπλισμένα με μοναδικά αναγνωριστικά και κατάλληλους μηχανισμούς ώστε να μπορούν να επικοινωνούν με άλλα αντικείμενα, χωρίς την ανάγκη ανθρώπινου παράγοντα. Αυτός ο κλάδος έχει αναπτύξει ιδιαίτερη δημοφιλία εξαιτίας μιας πληθώρας εφαρμογών, από Οικιακή Αυτοματοποίηση μέχρι τον χώρο της Υγείας. Μια πρόσφατη προσέγγιση στο ΔτΠ είναι το κοινωνικό Διαδίκτυο των Πραγμάτων(κΔτΠ), όπου τα αντικείμενα επικοινωνούν με βάση κοινωνικά χαρακτηριστικά.

Η ιδέα που ενέπνευσε αυτή την εργασία εντοπίζεται στην ανάγκη δημιουργίας μιας αποδοτικής ροής γνώσης στο κοινωνικό Διαδίκτυο των Πραγμάτων, μέσα από τον σχηματισμό κοινωνικών σχέσεων. Τα αντικείμενα α π οκτούν π ληροφορία μέσω αισθητήρων, η οποία μετασχηματίζεται σε μορφή γνώσης μέσω ανάλυσης και είναι ικανά να μάθουν και να επεκτείνουν τη γνώση τους, με βάση εμπειρίες άλλων αντικειμένων. Ο απώτερος στόχος αυτής της εργασίας είναι να διευκολύνει την αυτονομία στο κοινωνικό Διαδίκτυο των Πραγμάτων, προτείνοντας κατάλληλες σχέσεις αντικειμένων με σκοπό τη ροή γνώσης στο δίκτυο.

Αρχικά, παρέχεται στον αναγνώστη μια εισαγωγή στο βασικό υπόβαθρο και στη συνέχεια, γίνεται παρουσίαση του κυρίως θέματος της εργασίας αυτής. Το πρώτο κομμάτι ασχολείται με τις βασικές οντολογίες στο κοινωνικό Διαδίκτυο των Πραγμάτων: Εικονικές Οντότητες, Εφαρμογές και Χρήστες. Κάθε οντολογία ορίζεται α π ό ένα σύνολο χαρακτηριστικών τα οποία θα χρησιμοποιηθούν στη δημιουργία των προτεινόμενων Σχεσιακών Μοντέλων.

Έπειτα, αναλύονται τα δέκα Σχεσιακά Μοντέλα. Κάθε σχέση εξερευνάται σε βάθος, μελετώνας τα βασικά στοιχεία, απαιτούμενα για τη δημιουργία και στοιχεία από τη Θεωρεία Σχεσιακών Μοντέλων. Ακόμα, παρουσιάζονται κίνητρα και διάφορες προκλήσεις για κάθε σχέση, καθώς και αναλυτικά σενάρια εφαρμογής τους.

Τέλος, υλοποιείται μια προσομοίωση δικτύου, με τη χρήση της πλατφόρμας Gephi. Κάθε σχέση προσομοιώνεται ατομικά και γίνεται συζήτηση πάνω σε παρατηρήσεις και αποτελέσματα.

## Λέξεις Κλειδιά

Διαδίκτυο των Πραγμάτων, κοινωνικό Διαδίκτυο των Πραγμάτων, Σχεσιακά Μοντέλα, COSMOS, Εικονικές Οντότητες, ροή γνώσης, αυτονομία

## Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα την κα Θεοδώρα Βαρβαρίγου, η οποία ήταν και η επιβλέπων αυτής της εργασίας, για την ευκαιρία που μου έδωσε να δουλέψω πάνω στο θέμα αυτό και να μάθω περισσότερα πάνω στο συγκεκριμένο αντικείμενο, καθώς επίσης και τα μέλη της επιτροπής, κ. Βασίλειο Λούμο και κ. Συμεών Παπαβασιλείου.

Ακόμα, ένα μεγάλο ευχαριστώ οφείλω στον Ορφέα Βουτυρά, για τη στήριξη και βοήθεια που μου παρείχε καθ΄ όλη τη διάρκεια της εκπόνησης της διπλωματικής μου εργασίας. Οι εύστοχες παρατηρήσεις του και η προθυμία του για συζήτηση οποιαδήποτε στιγμή τη χρειάστηκα ήταν σημαντικά στοιχεία για την ολοκλήρωση της εργασίας μου.

Τέλος, θα ήθελα να ευχαριστήσω θερμά την οικογένεια μου, τον αδερφό μου και τους φίλους μου για τη διαρκή υποστήριξη τους κατά τη διάρκεια των σπουδών μου.

# CONTENTS

# Chapter 1: Introduction

## 1.1 Motivation

The Internet of Things (IoT) has been gaining ground in the past few years, blossoming from a novel concept to applicable system architectures. The idea behind the IoT is simple: enabling heterogenous devices to communicate with each other by exchanging information acquired through deployed sensors. The ultimate goal of the IoT is to create *smart objects*, able to construct and utilize an information network, as a way of becoming independent in making decisions. The rapid development of the IoT has spawned numerous surveys, protocols, white papers and suggested architectures on the subject, focusing on different approaches to the issue.

A recent approach to the Internet of Things has been the social Internet of Things(sIoT). In this approach, objects are able to socialize with other objects, creating various flows of social information in the network. The main objective of the sIoT is to separate the social network of humans from the network of things, by establishing a distinct social network of things. Humans are still able to control various aspects of this network, such as privacy, but objects will be responsible for making decisions, through sufficient information acquired through social communication.

One of the fundamental problems of the social Internet of Things is defining appropriate relationships among objects, in order to achieve scalable solutions. Objects need to be able to communicate efficiently; seek out other objects only when needed and in the least possible amount of steps, so as to avoid overloading the network. Data and information management mechanisms need to be provided to handle the exponentially increasing volume of digital data.

The focus of this thesis lies in proposing new Relational Models among objects in the social Internet of Things. Drawing from the human social network, similarities between social interactions of humans and social interactions of things are observed. By utilizing the Relational Models Theory that posits that people use four elementary models to generate, interpret, coordinate, contest, plan, remember, evaluate, and think about most aspects of most social interaction in all societies[1], we are able to propose ten Object Relationships to recreate social interactions in the network of objects. By expressing the social aspect of the Internet of things, all communications are enriched with a purpose and a scope. Objects will not attempt to blindly connect with all other objects in the network to acquire information, but will be able to aim at specific groups of objects based on the nature of the information instead.

## 1.2 Applications and Overview

As more and more objects and devices get involved in the Internet of Things, it is inherent that a socially enriched object network can have a plethora of applications. Major companies such as IBM, Amazon and Microsoft are currently heavily investing in the Internet of Things. Amazon recently introduced Dash Buttons, as a first step into Home Automation. Dash Buttons will allow users to order grocery items with the push of a WiFi-based connected button [2]. IBM has made a $3B investment to establish an Internet of Things unit inside of Big Blue along with a partnership with The Weather Company, to create Weather Data driven Applications [3]. Microsoft's Azure IoT services include Oil Supply solutions(Rockwell Automation)[4], Cloud Connected Elevators(ThyssenKrupp)[5] and an intelligent car-sharing service (Autolib')[6]. Although **Home Automation** and **Smart Cities** are currently the driving force of IoT applications, **Healthcare** is another area that can be enhanced by allowing devices and objects to communicate. MarketResearch.com reports that the healthcare Internet of Things market segment is poised to hit $117 billion by 2020[7].

This thesis is part of the COSMOS project (Cultivate resilient smart Objects for Sustainable city applicatiOnS)[8]. COSMOS is funded by the European Union and currently supports three scenarios:

• Smart heat and electricity management (London)
• Smart mobility for public transport (Madrid)
• IoT Business Eco-System (Taipei)

A brief description of each chapter is given below:

**Chapter 2: Background**

This chapter serves as an introduction to the prerequisites and the social aspect of this thesis. The main parts of this chapter include the Social Network, the Internet of Things, the social Internet of Things and the Cosmos Project. For each part, a brief description as well as a historic timeline is presented, followed by establishing the notion of knowledge flow, which is supported by appropriate examples.

**Chapter 3: Defining the Main Ontologies**

In this chapter, the three main ontologies of the COSMOS project are introduced. These ontologies include the Virtual Entities, Applications and Users. For each ontology, a basic description is provided, as well as definitions for the basic properties.

**Chapter 4: Analysis of the Proposed Relational Models**

This chapter contains the focus of this thesis, which is proposing Relational Models among VEs in the social Internet of Things. Ten Object Relationships are described. This description contains characteristics, required properties, necessity and motivation behind each Object Relationship, as well as Relational Model Theory elements, challenges and detailed examples. An hierarchy of these Object Relationships is also proposed, for two separate cases: fixed data and generic data.

**Chapter 5: Network Simulation and Results**

The final chapter of this thesis contains a network simulation of the proposed Relational Models. This includes metrics extraction and graph visualization for each Object Relationship, as well as observations and conclusions. Step by step analysis of the code used for the network generation is also presented.

# Chapter 2: Background

## 2.1 Social Networks

### 2.1.1 Brief Description of Social Networks

A social network is a social structure made up of a set of social actors (such as individuals or organizations) and a set of dyadic ties between them [9]. ***Actors*** are defined as the social entities among whom linkages are created. Actors are discrete individual, corporate, or collective social units. Examples of actors are people in a group, departments within a corporation, public service agencies in a city, or nation-states in the world system. ***Relational ties*** are the social ties that establish a linkage among actors. Some common examples of ties are:

- Evaluation by others( friendship, liking, respect)
- Association or affiliation ( jointly attending a social event, or belonging to the same social club)
- Behavioral interaction (talking together, sending messages)
- Biological relationship (kinship or descent)

A ***dyadic tie*** is the linkage or relationship between two actors. We can further define ***group*** as a collection of all *actors* on which ties are to be measured. A group consists of a finite set of actors who for conceptual, theoretical, or empirical reasons are treated as a finite set of individuals on which network measurements are made. Finally, a ***relation*** is the collection of *ties* of a specific kind among members of a *group*. For example, the set of friendships among pairs of employees in a company, or the set of formal diplomatic ties maintained by nations in the world, are ties that define relations. Various relations can exist for any group of actors.

Having defined *actor*, *group* and *relation*, we can conclude on a definition of a social network. A s***ocial network*** consists of a finite set or sets of actors and the relation or relations defined on them. The presence of relational information is a critical and defining feature of a social network.

### 2.1.2 Social Network Sites

Social Network Sites are web-based services that allow individuals to do the following [10]:

- Construct a public or semi-public profile within a bounded system
- Articulate a list of other users with whom they share a connection
- View and traverse their list of connections and those made by others
- Communicate and share information with other users

The first social network site launched in 1997. **SixDegrees.com** allowed users to create profiles, list their friends and surf the friends lists. Named after the *six degrees of separation* theory, in which any two people can be connected in a maximum of six steps, SixDegrees combined features found in community sites and chat clients such as AIM and ICQ, promoting itself as a tool to help people connect with and send messages to others. In **LiveJournal**, users can keep a blog, journal, or a diary. LiveJournal also offered one-directional connections, such that a user can "friend" another user without reciprocation. Users communicate by commenting on other users' journal entries.

**Friendster** was the first social network site to achieve mainstream popularity. Launched in 2002, it was designed to help friends of friends meet. Initially, the site restricted users from viewing profiles of people more than four degrees away. Following the success of Friendster, 2003 signaled a surge in social network sites. Maintaining the profile-centric design, these new social network sites aimed at particular groups of people. **LinkedIn** is aimed at business people and is mainly used for professional networking between job seekers, employees and employers, such as companies. Although not launched with bands in mind, **MySpace** became popular as a social network site for discovering music talent and connecting with musicians. Many media sites soon evolved into social network sites, such as **YouTube** for video sharing, **Last.FM** for music recommendation, as well as **Flickr** and **Pinterest** for photo sharing. Another noteworthy application which is based on social networks is **Foursquare**, where people can "check-in" in locations with their friends.

Undeniably, the most successful social network site of the 00s is **Facebook**. Initially designed to connect students from Harvard, the site soon expanded on other schools until it was eventually available to everyone. Users can add other users as friends, post status updates as well as other media such as videos, links and photos and communicate with other users through comments and messages. As of June 2014, Facebook had over 1.3 billion active users, with a revenue of $12.466 billions. **Twitter** was created in 2006. In Twitter, users can send and read messages called *tweets*, with a length restriction of 140 characters. Launched in 2010, **Instagram** enables users to upload photos and videos, while providing a variety of digital filters to add to these images.

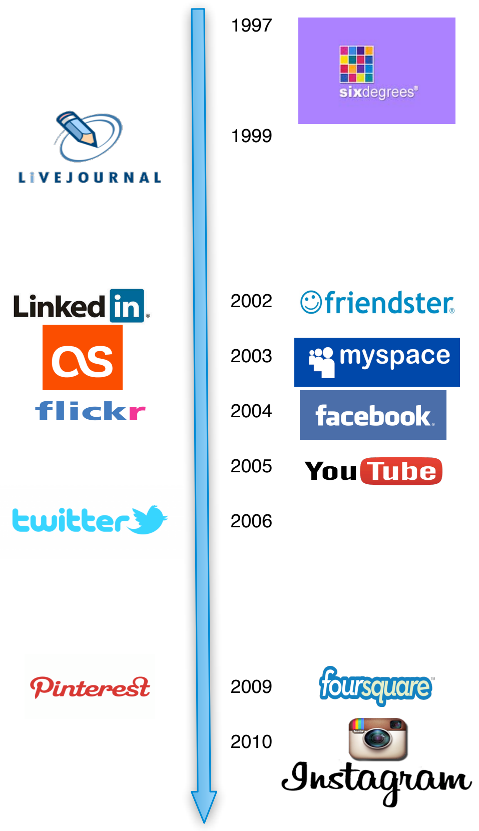


**Figure 2.1.2**: *Timeline of notable Social Network Sites*

### 2.1.3 Relational Models in Social Networks

The entities described in the definition of a Social Network can be found in all of the above social network sites. We can identify actors in the form of users of these sites. These users can be individuals or representing bigger groups of people, such as companies, organizations, or even nations. It is also interesting to note the variety of relations that can be identified among these sites. The most basic relation that can be found in one way or another in all of the sites, is the relation of **friends**. This relation is two-directional, as both users need to accept the request for this relation to be established. Friends are featured prominently in Facebook, which has also introduced another similar relation, called **close friends**. Close friends is aimed at creating a smaller group of friends, giving users the ability to filter information when their friends' list becomes too big. A variation of friends, named **connections**, can be found on LinkedIn, giving emphasis on professional networking.

The arrival of Twitter and Instagram popularized a new relation, called **followers**. In contrast to friends, this relation is asymmetric, meaning that a user can follow another user without needing approval and without expecting the other user to follow them back. The equivalent relation that is created with the introduction of followers is named **followees**. A user can have a number of followers, which are other users that follow him/her and that he/she may or may not be following back, as well as a number of users he/she follows. In that sense, he/she is a followee of the users he/she follows. Relationships can also be extracted by studying "check-ins" from location-based social networks, such as Foursquare [11]

### 2.1.4 Knowledge Flow in Social Networks

Social Network Sites provide a variety of available actions, which can be divided into two categories, a) posting information and b) interacting with information. Posting information is almost identical in most Social Networks. Users can post a "status message", which is simple text containing information, upload media such as photos and videos, or share links from the web. The content of the posted information can be sorted into the following categories:[12]

- Information Sharing
- Self Promotion
- Opinions/Complaints
- Statements and Random Thoughts
- Me now
- Question to followers/friends
- Presence Maintenance
- Anecdotes about the poster
- Anecdotes about others

The second category differs slightly from one Social Network to another. For example, Facebook users can like or comment a post, while Twitter users can retweet or favorite other tweets. However, we can spot two cases of interaction. The first one is to express approval for the content of the post and the second is to make the post known to other users of the social network. In both cases, a flow of information is created in Social Networks, through the creation of new content as well as interaction with existing content. Users have access to a plethora of information through their network of friends and followers. A general scenario of knowledge flow in Social Networks is the following: User

A has access to a type of information and wishes to make this information known to his/her extended social network, for example his/her friends on Facebook or his/her followers on Twitter. This information is then posted on a social network, in the form of plain text, web link or media, such as video,photo or audio. User B is part of User A's network and is notified of the post, as it is now visible. User B is has now acquired information that could be previously unknown to him/her, through User A. An example of this scenario is given below:

Alice is browsing the internet and she finds an article informing her of a strike in public transportation from 11:00 am to 04:30 pm. Using her mobile phone, she logs onto Facebook and posts the article with the caption: "No public transportation from 11:00-4:00. Guess I'll have to walk home". Bob has friended Alice on Facebook and can see her updates. Alice's post appears on Bob's timeline and he understands that the metro lines will be off. As Bob usually takes Line A to go to work, he now knows that he must alter his route. As this post proved useful to Bob, he likes it and shares the link saying: "Good to know! Was about to spend hours trying to figure my way out had I not seen this". Eve is not friends with Alice, yet she is friends with Bob. Eve is currently downtown and was planning on taking the metro through Main Street as well. Bob's post now appears on Eve's timeline and she is therefore informed that there will not be any metros coming through Main Street. She chooses to take her car instead.

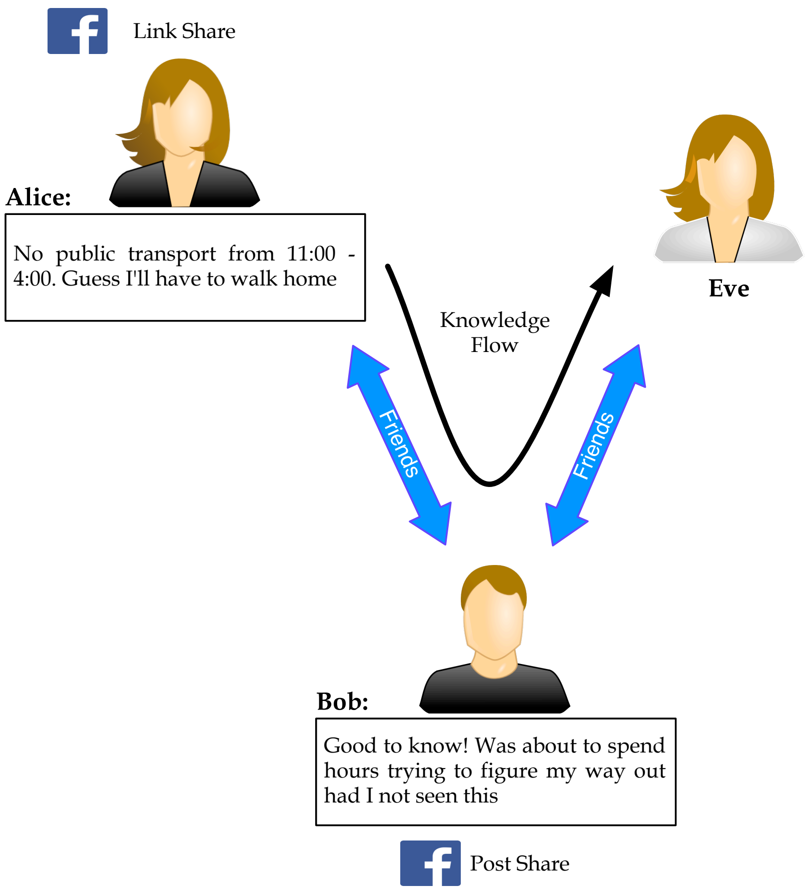

Link Share
Alice:
No public transport from 11:00 - 4:00. Guess I'll have to walk home
Eve
Knowledge Flow
Friends
Friends
Bob:
Good to know! Was about to spend hours trying to figure my way out had I not seen this
Post Share

***Figure 2.1.4***: *An example of Knowledge flow in Social Networks*

In the above scenario, we witnessed a knowledge flow from Alice to Bob and then to Eve. Both Bob and Eve were unaware of the information that Alice possessed, but through the use of social networks, they were both able to acquire new information that proved critical to their planning. Social Network Sites have proved influential in news spreading across the population, as studied on [13]. Extensive research has also been made in utilizing social networks in emergencies following disasters. In [14], data from social networks(Twitter) was gathered during the Colorado Floods in September 2013 to aid in remote reconnaissance work. Remote reconnaissance becomes critical when teams cannot get to the field, or cannot survey all areas of an affected area. Citizens' tweets, combined with satellite images and images taken with a DSLR camera on an Unmanned Aerial Vehicle were used to digitally survey a region and navigate optimal paths for direct observation.

## 2.2 Internet of Things

### 2.2.1 Brief History of Internet of Things

Early signs of the Internet of Things can be traced back to as early as 1932. Jay B. Nash writes in *Spectatoritis* [15]: "Within our grasp is the leisure of the Greek citizen, made possible by our mechanical slaves, which far outnumber his twelve to fifteen per free man… As we step into a room, at the touch of a button a dozen light our way. Another slave sits twenty-four hours a day at our thermostat, regulating the heat of our home. Another sits night and day at our automatic refrigerator. They start our car; run our motors; shine our shoes; and cult our hair. They practically eliminate time and space by their very fleetness." In 1946, a wristwatch named 2-Way Wrist Radio is introduced in the famous comic strip Dick Tracy, becoming one of the comic strip's most recognizable icons[16]. The 2-Way Wrist Radio facilitated communications between Tracy and members of the police force and is considered by many as an inspiration for smartwatches. In 1949, the bar code is conceived by Norman Joseph Woodland, when he drew four line in the sand on a Miami beach [17]. This led to the invention of the Universal Product Code (UPC), the ubiquitous bar code used in supermarkets. In 1967, Hubert Upton invents an analog wearable computer with eyeglass-mounted display to aid in lip reading [18]. In 1969, the first message is sent over the ARPANET, the predecessor of the Internet [19].

In 1973, Mario Cardullo received the first patent for a passive, read-write RFID tag [20]. In the early 1980s, members of the Carnegie-Mellon Computer Science department installed micro-switches in the Coke vending machine so they could see how many bottles were present in the machine and whether they were cold or not [21]. In 1991, Xerox PARC's Mark Weiser publishes "The Computer in the 21st Century" in *Scientific American*, where the terms of "ubiquitous computing" and "embodied virtuality" were first used [22]. In 1994, Xerox EuroPARC's Mik Lamming and Mike Flynn demonstrate the Forget-Me-Not, a wearable device that records interactions with people and devices, storing the information in a database [23]. In September of the same year, the term "context-aware" was first used by B.N. Schilit and M.M Theimer in "Disseminating active map information to mobile hosts," Network, Vol. 8, Issue 5 [24].

In October 13-14, 1997, Carnegie-Mellon, MIT and Georgia Tech co-hosted the first IEEE International Symposium on Wearable Computers, in Cambridge, MA. In 1999, the Auto-ID (for Automatic Identification) Center is established at MIT. Sanjay Sarma, David Brock and Kevin Ashton turned RFID into a networking technology by linking objects to the Internet through the RFID tag [25]. Two years later, the Auto-ID Center would introduce a new object identification scheme, the Electronic Product Code (EPC). In 2005, a team of faculty members at the Interaction Design Institute Ivrea in Ivrea, Italy, developed

Arduino, a cheap and easy to use single board micro-controller. Arduino made a huge impact on the world of physical computing.

### 2.2.2 Definitions of Internet of Things

The Internet of Things (IoT) is a novel paradigm that is rapidly gaining ground in the scenario of modern wireless telecommunications. The basic idea of this concept is the pervasive presence around us of a variety of things or objects – such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals [26]

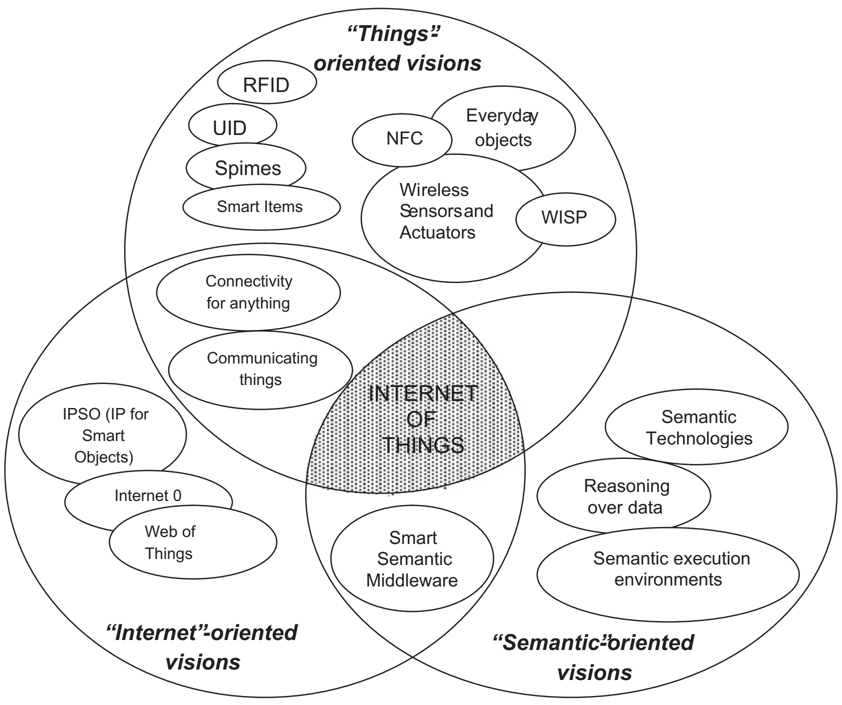
We can identify three major different definitions that arise from the name "Internet of Things" [27]:

- Things Oriented Perspective
- Internet Oriented Perspective
- Semantics Oriented Perspective

The very first definition of IoT derives from a "Things oriented" perspective; the considered things were very simple items: Radio-Frequency IDentification (RFID) tags [28]. Although RFID is still the leading technology in the Things Oriented perspective, due to maturity, low cost and strong support from the business community, a wide portfolio of device, network and service technologies are currently building up the IoT, such as Near Field Communications (NFC) and Wireless Sensor Actuator Networks(WSAN).

On the other hand, the Internet Oriented Perspective calls for a global infrastructure which connects both virtual and physical generic objects and highlights the importance of including existing and evolving Internet and network developments[29]. An example of the Internet Oriented Perspective can be found at IP for Smart Objects (IPSO), a forum formed in September 2008 to promote Internet Protocol as a network technology for connecting Smart Objects around the world. Justification for the IPSO lies in the idea that the IP stack is a light protocol that already connects a huge amount of communicating devices and runs on tiny and battery operated embedded devices. This guarantees that IP has all the qualities to make IoT a reality.

Finally, the idea behind the Semantic Oriented Perspective is that the number of items involved in the Future Internet is destined to become extremely high, thus creating issues regarding organization, search, representation and storage of IoT information [30]. Semantic technologies can exploit appropriate modeling solutions for things description, reasoning over data, semantic execution environments and architectures, as well as scalable storing and communication infrastructure. The following figure gives an outline of the Internet of Things, as a combination of these three perspectives.



"Things"-oriented visions
RFID
UID
Spimes
Smart Items
NFC
Everyday objects
Wireless Sensors and Actuators
WISP
Connectivity for anything
Communicating things
INTERNET OF THINGS
IPSO (IP for Smart Objects)
Internet 0
Web of Things
Smart Semantic Middleware
Semantic Technologies
Reasoning over data
Semantic execution environments
"Internet"-oriented visions
"Semantic"-oriented visions

***Figure 2.2.2****: Internet of Things as a result of three different visions* [27]

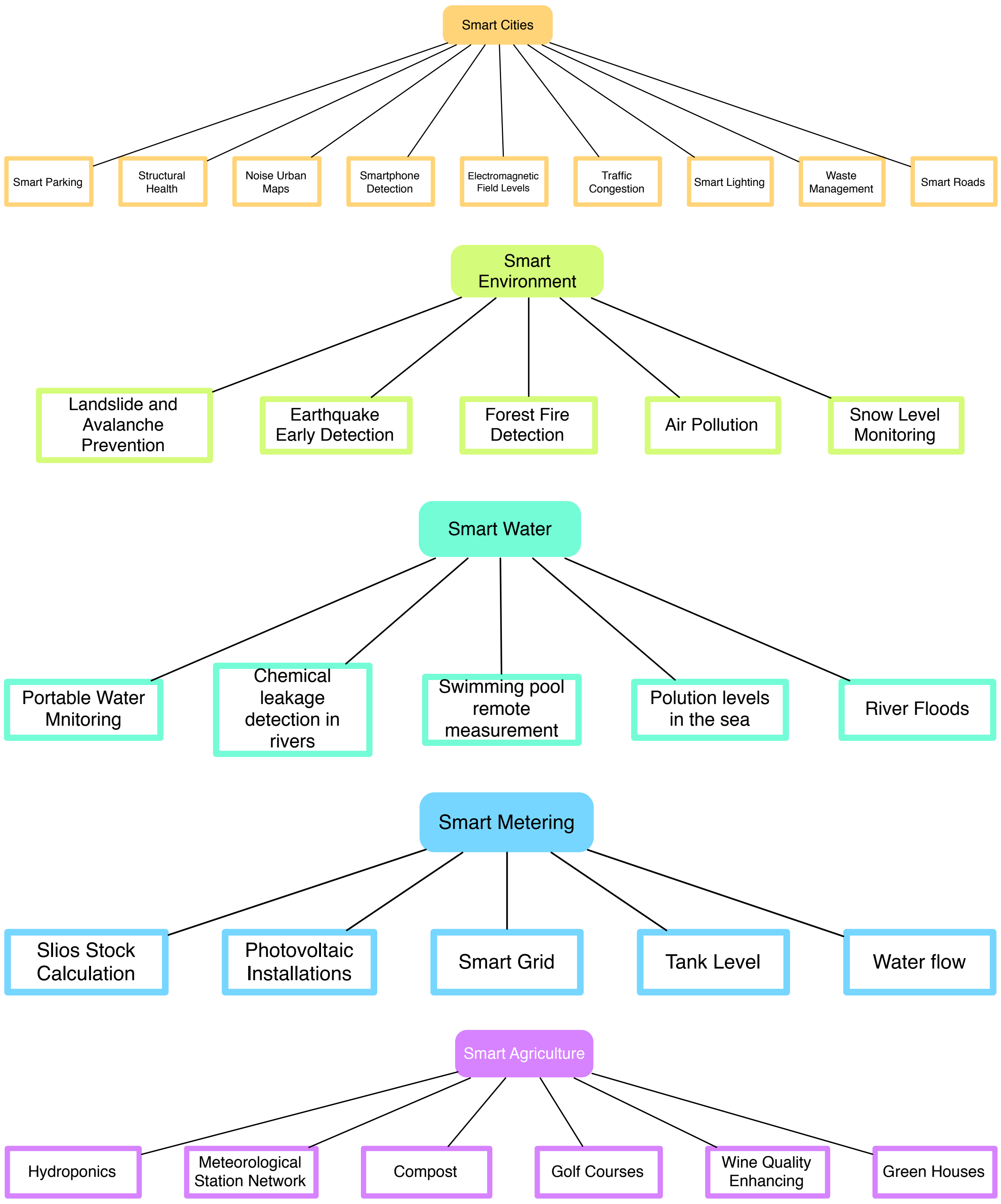
## 2.2.3 Internet of Things: Technologies

The most common technologies used in building the Internet of Things are RFID systems and Wireless Sensor Networks.
A Radio Frequency Identification (RFID) system consists of RFID readers and RFID tags. Each tag has a unique identifier and can be applied to objects [31]. RFID readers trigger the tag transmission by generating an appropriate signal, which essentially searches for tags in the vicinity. Objects do not need to be in line-of-sight for the pairing of a tag and a reader to happen. RFID tags can be *passive*, meaning that they do not rely on-board power supplies, but are powered from signals by a RFID reader nearby. RFID tags can also use power supplies, such as batteries, making them *semi-passive*, or active. In semi-passive RFIDs, batteries power the microchip while receiving the signal from the reader (the radio is powered with the energy harvested by the reader signal). Differently, in active RFIDs the battery powers the transmission of the signal as well. Obviously, the radio coverage is the highest for active tags even if this is achieved at the expenses of higher production costs.

Sensor networks can cooperate with RFID systems to better track the status of things, such as their location or temperature, acting as a bridge between the physical and the digital world. Sensor networks consist of a certain number of sensing nodes communicating in a wireless multi-hop fashion. Usually nodes report the results of their sensing to a small number (in most cases, only one) of special nodes called *sinks*. Design objectives of the proposed solutions are energy efficiency, scalability (the number of nodes can be very high), reliability (the network may be used to report urgent alarm events), and robustness (sensor nodes are likely to be subject to failures for several reasons) [32].

### 2.2.4 Domains

The Internet of Things can, or is already applied to a variety of domains. These domains are separated in the following categories [33]:



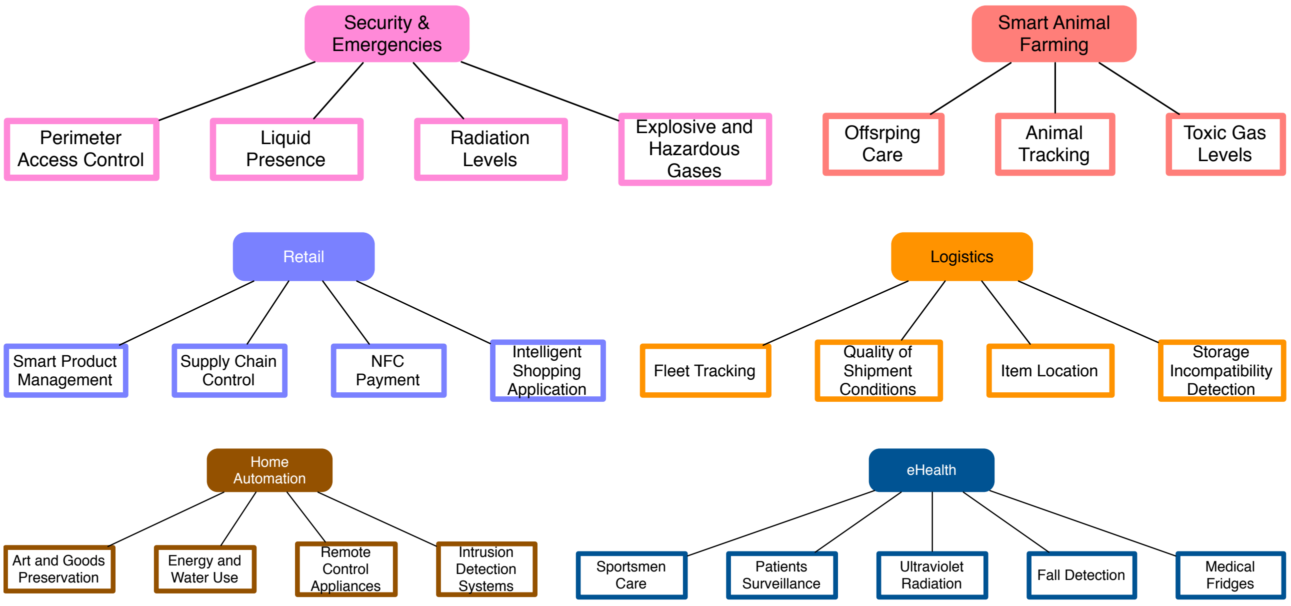


***Figure 2.2.4**: Available domains, separated in categories*

### 2.2.5 Open Issues

A number of open issues regarding the Internet of Things can be found in [27]. These include:

• Standardization
• Mobility Support
• Naming
• Transport Protocol
• Traffic Characterization and QoS Support
• Authentication
• Data Integrity
• Privacy
• Digital Forgetting

Issues regarding *standardization* stem from the fact that there are several standards so far, yet no integration in a comprehensive framework has been achieved yet. There are also many proposals for object addressing but none for *mobility support* in the IoT scenario, where scalability and adaptability to heterogeneous technologies represent crucial problems. As far as *naming* goes, Object Name Servers (ONS) are needed to map a reference to a description of a specific object and the related identifier, and vice versa.

Existing *transport protocols* fail in the IoT scenarios, since their connection setup and congestion control mechanisms may be useless; furthermore, they require excessive buffering to be implemented in objects. The IoT will generate data traffic with patterns that are expected to be significantly different from those observed in the current Internet. Accordingly, it will also be necessary to define new QoS requirements and support schemes.

*Authentication* is difficult in the IoT as it requires appropriate authentication infrastructures that will not be available in IoT scenarios. Furthermore, things have scarce resources when compared to current communication and computing devices. Also man-in-the-middle attack is a serious problem. *Data integrity* is usually ensured by protecting data with passwords. However, the password lengths supported by IoT technologies are in most cases too short to provide strong levels of protection

A lot of *private* information about a person can be collected without the person being aware. Control on the diffusion of all such information is impossible with current techniques. Finally, regarding *digital forgetting*, all the information collected about a person by the IoT may be retained indefinitely as the cost of storage decreases. Also data mining techniques can be used to easily retrieve any information even after several years.

### 2.2.6 Knowledge Flow in the Internet of Things

The Internet of Things has made knowledge flow between things possible, without the need of a user. The use of wireless sensors, as well as RFID systems has enabled a variety of applications, ranging from home automation to smart cities and eHealth. In this section, a few examples of IoT applications are presented, along with a brief description of knowledge flow as well as scenarios of use.

*Nest Learning Thermostat (Home Automation)*

Nest Labs, a home automation company in Palo Alto, California, introduced the Nest Learning Thermostat in 2011, a self-learning Wi-Fi enabled thermostat that is able to optimize heating and cooling to conserve energy. Nest Thermostat is designed to program itself, by learning its user habits and preferences. A noteworthy feature of Nest Thermostat is its Auto-Away function. Using a combination of sensors and algorithms, the thermostat is able to notice when a user is away from home or when he/she comes back [34]. This is used to turn itself down when everyone is away and turn itself back up when someone arrives home. Nest Thermostat can also be controlled remotely, by use of mobile applications. Nest Thermostat was so successful, that it led to Nest Labs's acquisition by Google for $3.2 Billion on January 14, 2014 [35].

*Philips Hue Lamps (Home Automation)*

Philips introduced its new LED lighting system, Hue, in 2012. This new lighting system allows light bulbs to be controlled and customized with the use of a mobile application. Users can program a bulb to notify them of various activities, such as emails or appointments. Users can customize the hues and the lighting in a room with modes such as "Reading", "Relax", "Energize" and more. Using geofencing technology, Hue can also detect when a user is back home, to turn the lights on.

*Mimo Smart Baby Monitor ( eHealth)*

The Mimo Smart Baby Monitor utilizes a combination of sensors that relay information via Bluetooth through the cloud to a connected device [36]. This Baby Monitor can relay a variety of information, such as sleep status, baby breathing and body position. Parents can also listen in on their babies, with live audio transmitted through the monitor, using a mobile device. Statistics about baby activity are gathered with the use of sensors as well.

A scenario using the above three applications can be the following. Alice has installed the Nest Learning Thermostat as well as the Philips Hue lighting system at her home, as well as the accompanying applications on her phone. Alice has also equipped her daughter, Eve, with a Mimo Smart Baby Monitor. Eve is currently staying with her father, Bob. Once

Alice comes home, the Nest Thermostat, as well as the Philips Hue lighting system can detect her arrival, turning the thermostat up and the lights on. A light bulb goes off, notifying Alice not to forget her doctor appointment in one hour. Alice uses her phone to see that her daughter Eve is not sleeping and is having a strange body position. Worried, she phones Bob, who then checks up on Eve to find her ready to jump off her crib.

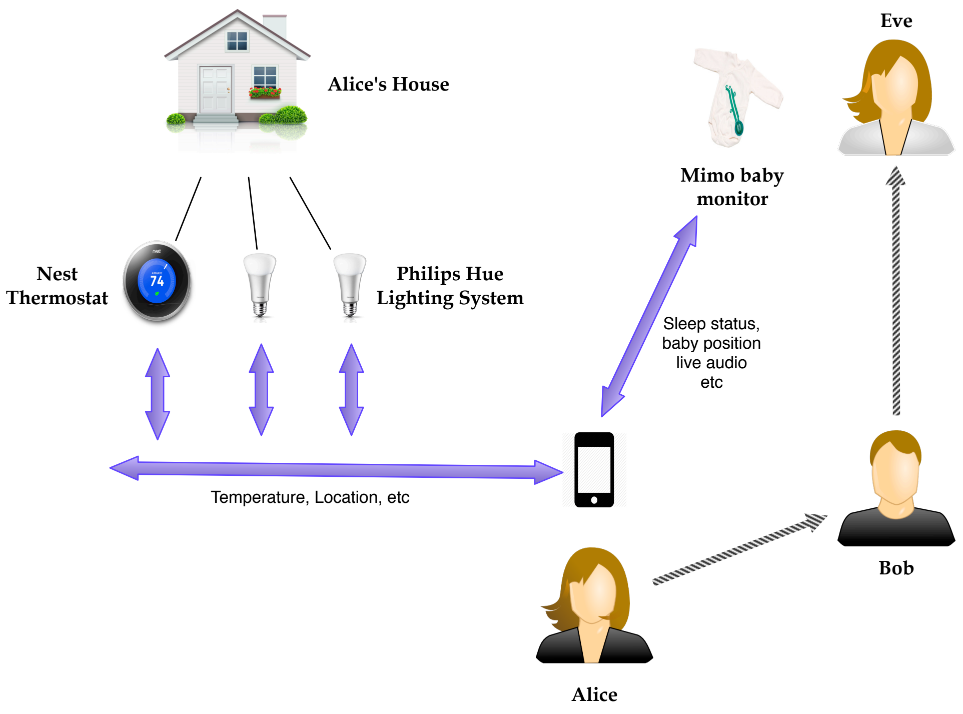

Alice's House
Eve
Mimo baby monitor
Nest Thermostat
74
Philips Hue Lighting System
Sleep status, baby position live audio etc
Temperature, Location, etc
Bob
Alice

***Figure 2.2.6.1**: Knowledge Flow in IoT scenario 1*

In the above scenario, we can notice two types of knowledge flow. We have knowledge flow between things, with information regarding room temperature, geolocation, sleep status, baby position etc, flowing between the Nest Thermostat, the Philips Hue Lighting System, the Mimo Baby Monitor and Alice's mobile device. This flow makes feasible a secondary flow, between users Alice, Bob and Eve.

The Internet of Things has been used in larger scale applications as well, in cases of waste management, smart parking and smart city lighting, for example. Bigbelly provides a smart, self-powered waste management & recycling solution, with waste and recycling stations that can monitor and report station fullness remotely [37].

Streetline gathers information regarding parking vacancies in parking lots and streets to provide real time parking guidance as well as statistics regarding parking activity to provide a better picture on what is going on out on the streets [38].

Echelon provides Wired and Wireless Outdoor Lighting systems that can be controlled to reduce energy consumption, improve safety for both pedestrians and drivers as well as serve in a range of other IoT applications. Echelon's power line communications solutions are in use at over 600 cities and locations worldwide, with case studies in Oslo, Norway [39] and in Sénart en Essonne, France [40].

The above applications can be used by a municipality for example, to better track the city's streets and traffic, cut down on energy consumption with smart lighting control and maintain a clean and habitable environment for citizens.

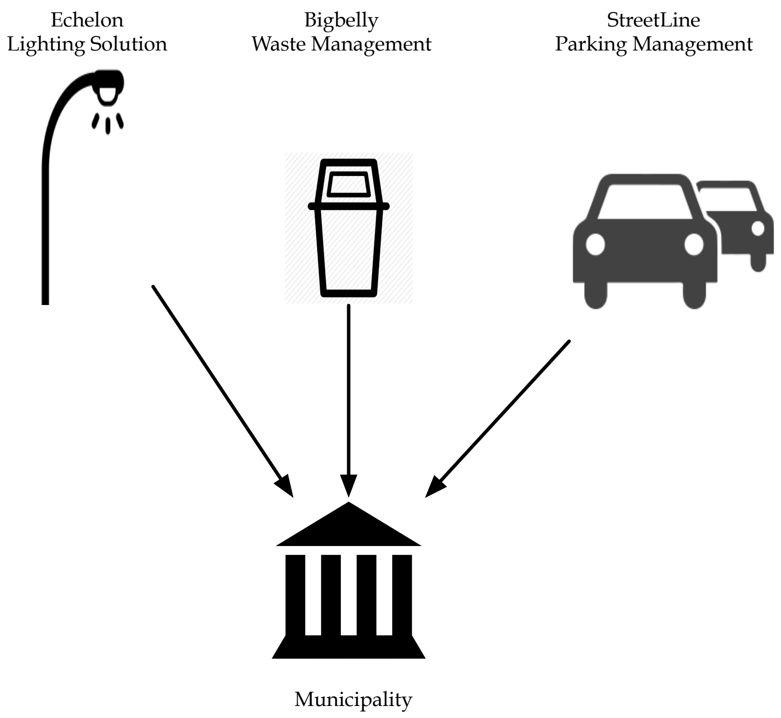


***Figure* 2.2.6.2**: *Knowledge Flow in IoT scenario 2*

## 2.3 Social Internet of Things

### 2.3.1 From the Internet of Things to the social Internet of Things

The basic idea behind the social Internet of Things is the definition of a "social network of objects", analogous to the Social Networks Sites for humans. The possible advantages are [41]:

- Give the IoT a structure that can be shaped as required to guarantee network navigability, so as that object and service discovery is effectively performed and scalability is guaranteed like in human social networks
- Extend the use of models designed to study social networks to address IoT related issues (intrinsically related to extensive networks of interconnected objects).
- Create a level of trustworthiness to be used for leveraging the level of interaction among things that are "friends".

A first idea of socialization between objects was introduced in [42], as part of the Smart-Its project on technologies for computer-augmentation of everyday artifacts. The object was to develop a range of small, embedded devices as platforms for augmentation and interconnection of these artifacts. These devices, named Smart-Its, would integrate sensing, processing and communication with variations in perceptual and computational capability, in order to facilitate autonomous awareness of an artifact's context, independent of infrastructure. The work on [42] focused on enabling these smart devices to establish temporary relationships, with the mediation of their owners.

In [43], a new type of objects is introduced, called Blogject, meaning "objects that blog". These objects can participate within the Internet of social networks, as opposed to things simply connected to the Internet. Embodied Microblogging is introduced in [44], for the design of digital technology for social interaction. Embodied Microblogging is defined as informing the design of augmented objects of everyday life, for facilitating senior citizens's everyday life. In the paper, Walky is introduced as a way of Embodied Microblogging. Walky combines rollators with sensors and dedicated displays. Using the rollator causes a broadcast of walking activity to the user's community. Senior citizens can, with the use of Walky, have an idea of friends around them, enabling them to interact with others, such as meeting for a walk.

In [45], research is focused on enabling things to participate with humans in forming a socio-technical network. Using Twitter, the authors enable things to start communicating their status and functionality, by sharing pictures, comments and sensor data via social networks. Finally, in [46], objects can develop a networking infrastructure based on the information that is spread, rather than information on the objects themselves.

However, in order to achieve autonomy in the social Internet of Things, three main points need to be made:

- Social relationships need to be established among things, without the need of their owners. Owners can participate, to some extent, to the objects relationships, however, it is the objects that need to play a key role in this scenario.
- The establishment of social relationships among objects can allow a more effective and efficient way for objects to discover services and resources, without the presence of humans.

- The envisioned IoT architecture is not a mere service platform centered on the concept of web of things, yet a real platform for SNSs with suitable components introduced to cope with the presence of objects instead of human beings.

### 2.3.2 Evolution from Smart Objects to Social Objects

In the Internet of Things, every human and every object has a locatable, addressable and readable counterpart in the Internet. In this scenario, objects can produce and consume services and collaborate with other counterparts towards a common goal. This is possible, thanks to intense interactions among objects, which collaborate to realize complex services. This has brought the design of new generations of "smart objects" able to discover new services, make new acquaintances, exchange information, connect to external services as well as collaborate towards a common goal[47]. Two important questions that are currently being investigated are the following:

- Are smart objects expected to manifest new potentials?
- Can these potentials lead to a fully networked human society by introducing more effective IoT models?

In [47], a parallel between human (and animal) evolution and object evolution is introduced. In the natural world, examples are ample concerning animals and humans that managed to overcome the difficulties of their complex environments by creating groups and forming social relationships. In this light, we can consider a new generation of social objects that:

a) can achieve an autonomous interaction with other objects, without the mediation of their owners;
b) can discover services and information in a trust-oriented way by searching the Internet of Things
c) can inform the rest of the network of their presence.

In analogy with the human evolution from *homo sapiens* to *homo agens,* a similar evolutionary path from a *res sapiens* (smart object) to a *res agens* (acting object), an object capable of acquiring knowledge and converting it into actions, can be made. This can lead to a new type of object, *res socialis* (social object), an object that has the ability to take part in a social community along with other objects.

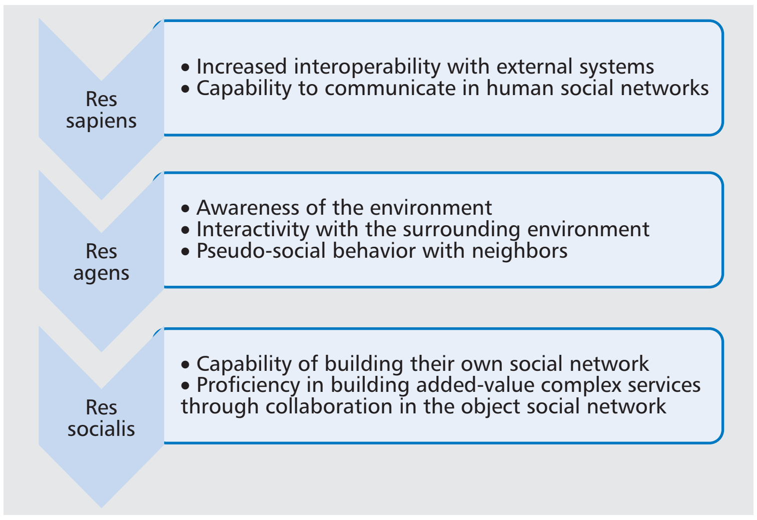

Res sapiens
• Increased interoperability with external systems
• Capability to communicate in human social networks
Res agens
• Awareness of the environment
• Interactivity with the surrounding environment
• Pseudo-social behavior with neighbors
Res socialis
• Capability of building their own social network
• Proficiency in building added-value complex services through collaboration in the object social network

***Figure 2.3.2****: The path from Res Sapiens to Res Socialis* [47]

### 2.3.3 The sIoT Architectural Model

In [41], an sIoT Architectural Model is proposed, in comparison to a common architectural model for Social Network Sites for humans.

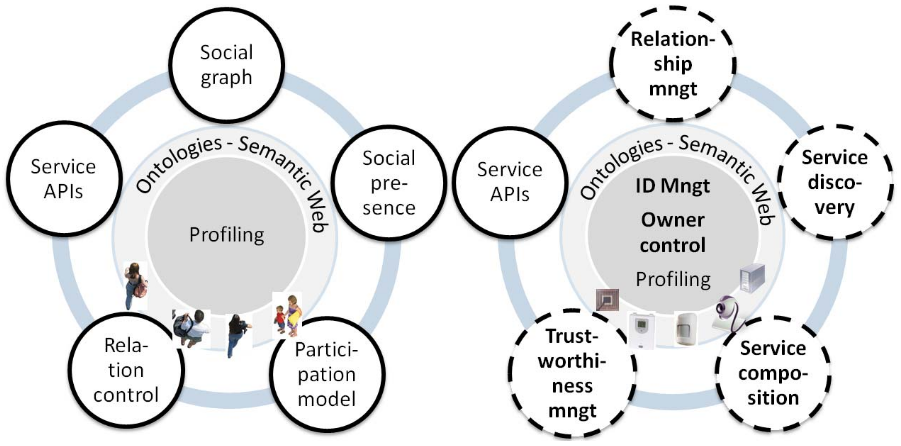


***Figure 2.3.3.1***: *Basic components of Social Networks for humans(left) and objects(right)* [41]

In the architecture for objects, we can distinguish three basic components:

• ID Management (ID)
• Object Profiling (OP)
• Owner Control (OC)

The *ID Management* is responsible for assigning appropriate, unique IDs to objects. This system could be a combination of IPv6 addresses, Universal Product Codes (UPC), Electronic Product Codes (EPC) or other technologies. The *Object Profiling* contains static and dynamic information about the object. This allows object organization into classes, based on main object features. The Owner Control contains information and policies defined by the owner, regarding the object's behavior. The owner is responsible for deciding what information can be shared and what relationships and with which objects they can be created. It is understood that this component is in need of existing security and access control policies.

The satellite components of this architecture are the following:

• Relationship Management (RM)
• Service Discovery (SD)
• Service Composition (SC)
• Trustworthiness Management (TM)
• Service APIs

*Relationship Management* contains the "intelligence" that allows objects to start, update and terminate relationship. This component is the main focus of our research. *Service Discovery* is responsible for querying the social relationship network to find objects that can provide services or knowledge. Service Discovery is the analogous of Social Presence in Social Network Architecture, where humans seek friends and information. *Service Composition* is the replacement of the Participation Model. It utilizes object relationships to find and activate a service. Two approaches to service composition are suggested, the *reactive* and the *proactive* approach. Crowd information processing will also be included in this component, to allow the object to process acquired information to choose the most reliable answer. *Trustworthiness Management* is needed to establish reliability of the acquired information, based on the behavior of the object. This component is strictly related to the RM component. Various aspects from the social networks can be used in calculating trustworthiness, such as centrality and prestige. Finally, the *Service APIs* component is analogous to the one required in the social networks architecture model. This contains all the APIs needed to enable the object to perform services.

Based on these components, the social Internet of Things architecture on the server side is comprised of three layers:

• The Base Layer
• The Component Layer
• Application Layer

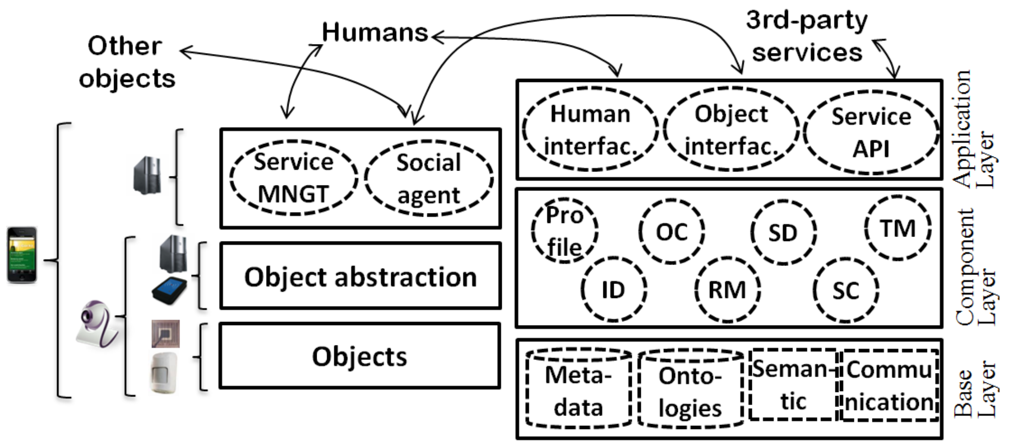


***Figure 2.3.3.2***: *Architecture of the sIoT: Client Side(left) and server side(right)* [41]

The *Base layer* includes the needed databases and engines. This layer includes the database for storage and management of data with relevant descriptors, the ontologies database and semantic engines and communications. The *Component layer* contains tools for implementing all components. Interfaces to objects, humans, and third-party services are in the *Application layer*.

On the object side, the first architectural layer – named the Object layer – is where the physical objects are located and are reached through their specific communication interfaces. An Object abstraction layer is thus needed to harmonize the communication of the different devices through common languages and procedures. In the case of elementary objects, such as an RFID-tagged object, a gateway is required to implement this abstraction layer, while for more complex objects this layer can be implemented in the object itself. In the third layer, the Social agent is devoted to the communication among objects and with SIoT servers to update profile and friendships and to discover/request services from the social network. Finally, the Service management is the interface of humans to control the object behavior in the SIoT.

### 2.3.4 Relational Models Theory

As mentioned before, the basic idea behind the social Internet of Things is the definition of a "social network of objects", analogous to the Social Networks Sites for humans. In order to achieve that, we need to establish social relationships among objects. The focus of this research lies in identifying proper object relationships that will allow the objects to achieve autonomy, by discovering and sharing services and resources on their own.
In order to identify these relationships, we turned to Sociology, Anthropology and Cognition studies. Alan Fiske's Relational Models Theory provided helpful insight in this search. According to Fiske, human relationships and social systems are culture-specific implementations of just four elementary relational models in various combinations. These models are presented as [48]:

*Communal Sharing (CS)*

Communal Sharing is an equivalence relation, in which people attend to something important they have in common. People in each group are the same in respect to the matter at hand; outsiders are different. Distinguishing individual identities are socially irrelevant. Generosity within a Communal Sharing group is not usually conceived of as altruism due to this shared identity, even though there is typically much behavior which otherwise would seem like extreme altruism [49]. Examples of Communal Sharing include nationalism, racism, intense romantic love, indiscriminately killing any member of an enemy group in retaliation for the death of someone in one's own group or sharing a meal.

*Authority Ranking (AR)*

Authority Ranking is a linear hierarchy in which people are asymmetrically differentiated in the current context. The higher ranked enjoy prestige and privilege and typically have some control over the lower ranked. However, it is possible for the higher ranked to have duties of protection and affection for those beneath them. Examples include kings and princes, military rankings, parents and children, case systems and God's authority over humankind. However, it must be stated that manipulation is not considered Authority Ranking, but is categorized as the Null Relation in which people treat others in non-social ways [49].

*Equality Matching (EM)*

Equality Matching is a relationship in which people keep track of additive differences, with even balance as the reference point. When perfect balance is not maintained, people calculate how much correction is needed. Examples include the principle of one-person/ one-vote, rotating credit associations, equal starting points in a race, taking turns offering dinner invitations, and giving an equal number of minutes to each candidate on debates [49].

*Market Pricing (MP)*

Market pricing is based on a socially meaningful proportionality, where the ratio may concern monetary value, utility, efficiency, effort, merit, or anything else. In Market Pricing, all socially relevant properties of a relationship are reduced to a single measure of value, such as money or pleasure. Most utilitarian principles involve maximization. An exception would be Negative Utilitarianism whose principle is the minimization of suffering. But all utilitarian principles are applications of Market Pricing, since the maximum and the minimum are both proportions. Other examples include rents, taxes, cost-benefit analyses including military estimates of kill ratios and proportions of fighter planes potentially lost, tithing, and prostitution [49].

These four basic relational structures can be applied to objects as well. In Communal sharing relationships for example, equivalence and collectivity membership emerge against any form of individual distinctiveness. These can be definitely associated with behaviors of objects that have collective relevance only. This is, for example, the case of "swarms" of objects for which is only important the service that the whole swarm can provide to users [41]. Equality matching can characterize the information exchange among equal objects. Authority Ranking can be viewed in relationships between different levels of information exchange, for example RFID readers and tags or Bluetooth master and slave devices. Finally, Market pricing can be viewed as objects forming relationships in order to gain something. The basis of these relationships is, therefore, mutual benefit.

## 2.4 COSMOS

### 2.4.1 What is COSMOS

The COSMOS project (Cultivate resilient smart Objects for Sustainable city applicatiOnS) aims at enhancing the sustainability of smart city applications, by allowing IoT based systems to reach their full potential [8]. The project's goal is to achieve autonomous evolution of things, making them more reliable and smarter. In order to achieve that, things will be able to learn based on others experiences, while situational knowledge acquisition and analysis will make things aware of conditions and events potentially affecting their behavior. Management decisions and runtime adaptability will be based on things security, trust, administrative, location, relationships, information and contextual properties. Data and information management mechanisms are available by COSMOS to handle the exponentially increasing "born digital" data.

COSMOS enables smart city IoT applications to take full advantage of its technologies, through three representative scenarios:

- Smart heat and electricity management (London)
- Smart mobility for public transport (Madrid)
- IoT Business Eco-System (Taipei)

### 2.4.2 MAPE-K Model

In order to achieve self management and autonomy, COSMOS follows the MAPE-K model. Introduced by IBM in 2006, the MAPE-K loop is used as an autonomic manager. An autonomic manager is an implementation that automates some management function and externalizes this function, according to behavior defined by management interfaces [50]. The autonomic manager is a component that implements an intelligent control loop. For a system component to be self-managing, it must have an automated method to collect the details it needs from the system; to analyze those details to determine if something needs to change; to create a plan, or sequence of actions, that specifies the necessary changes; and to perform those actions. When these functions can be automated, an intelligent control loop is formed.

The MAPE-K loop consists of four parts that share knowledge:

• The **monitor** function
• The **analyze** function
• The **plan** function
• The **execute** function

The *monitor* function provides mechanisms needed to collect, aggregate, filter and report details collected from a managed resource. The *analyze* function provides the mechanisms that correlate and model complex situation such as time-series forecasting. These mechanisms allow the autonomic manager to learn about the IT environment and help predict future situations. The *plan* function provides the mechanisms that construct the actions needed to achieve goals and objectives, by using policy information to guide its work. Finally, the *execute* function provides the mechanisms that control the execution of a plan with considerations for dynamic updates. Autonomic managers provide sensor and effector interfaces that can be used by other autonomic managers and other components in the distributed infrastructure. Using these sensor and effector interfaces enables these components to be composed together in a manner that is transparent to the managed resources.

A **knowledge source** is an implementation of a registry, dictionary, database or other repository that provides access to knowledge according to the interfaces prescribed by the architecture. In an autonomic system, knowledge consists of particular types of management data with architected syntax and semantics, such as symptoms, policies, requests for change, and change plans. This knowledge can be stored in a knowledge source so that it can be shared among autonomic managers.

The knowledge stored in knowledge sources can be used to extend the capabilities of an autonomic manager. An autonomic manager can load knowledge from one or more knowledge sources, and the autonomic manager's manager can activate that knowledge, allowing the autonomic manager to perform additional management tasks (such as recognizing particular symptoms or applying certain policies). Data used by the autonomic manager's four functions (monitor, analyze, plan, and execute) are stored as knowledge that could be shared among autonomic managers. The knowledge includes data such as topology information, historical logs, metrics, symptoms, and policies.

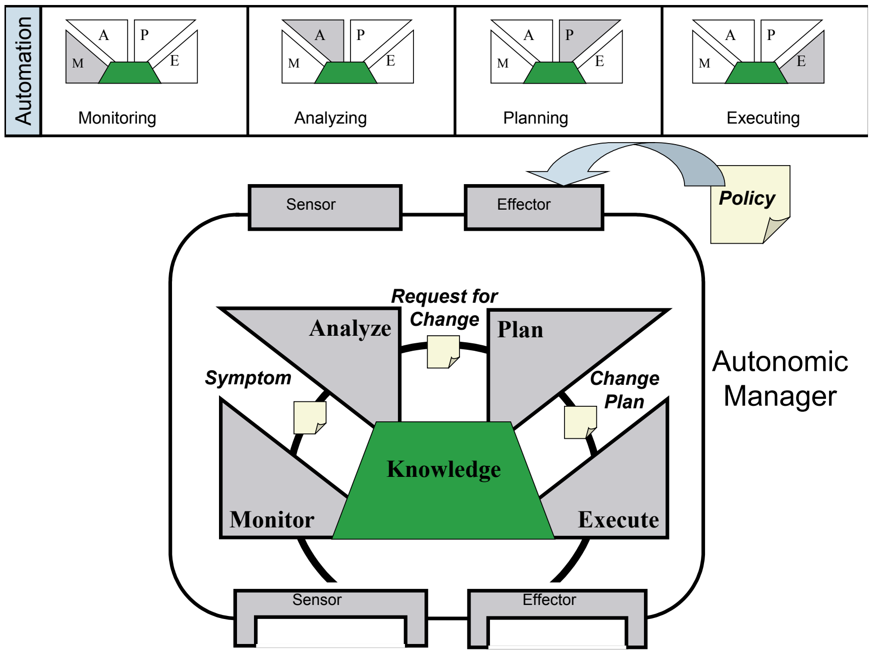


***Figure* 2.4.2.1**: *An autonomic manager* [50]

The MAPE-K loop can get all the data needed to achieve an autonomous cycle from the IoT and at the same time provide to the IoT optimal self managing functionalities. However, the social approach to the Internet of Things calls for an extension of the MAPE-K loop model. Two new component are introduced: Social Monitoring (SM) and Social Analysis (SA) [51].

The **Social Monitoring** component contains all the main tools and techniques that are used for the monitoring of the social properties of the things, such as Trust and Reputation. This component collects, aggregates and distributes monitoring data, in the form of events, across the decision making components of the collaborating groups. The **Social Analysis** component is used for the extraction of complex social characteristics, as well as models and patterns regarding the behavior and the relations between things. Social Network Analysis is used in this component.

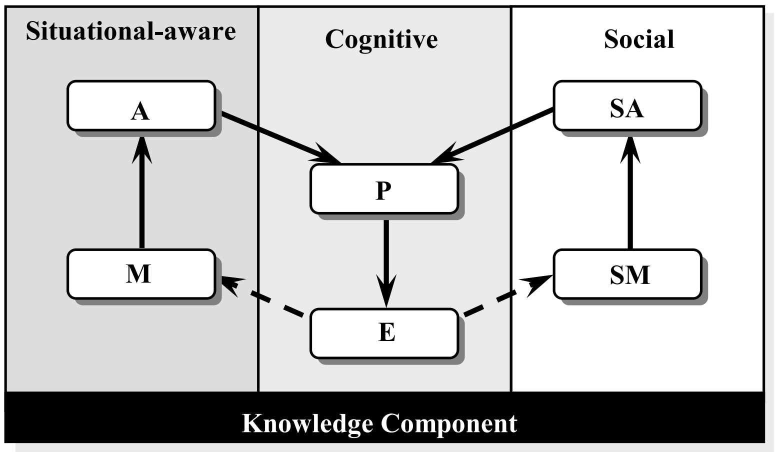


***Figure* 2.4.2.2**: *The COSMOS MAPE-K loop* [52]

## 2.4.3 Knowledge Management: The DIKW Pyramid

The Internet of Things will cause a major flood of real world information to the virtual world. The use of sensors, connected by computer systems, software and services, will make things more and more aware of what happens in the real world, in real time, everywhere. Therefore, there is a need for data and information management mechanisms to efficiently handle the exponentially increasing "digital born" data. This transformation from raw data into knowledge is one of the biggest challenges behind the Internet of Things. There is an entire cycle of data processing up to the generation of cooperative knowledge networks. These knowledge networks can feed complex hierarchical feedback control loops, since sensorial data is very important for decision making. Decisions made on the virtual side can be reflected on the real environment helping us to better use our resources. Hence, a first step to designing the general architecture of a project on the IoT domain and realizing its capabilities and chances for evolution is the definition of its own Knowledge Management (KM) cycle [51].

Knowledge management is the process of capturing, developing, sharing and effectively using knowledge and summarizes all activities with the goal of using knowledge in a more efficient and effective manner, achieving certain objectives. A Knowledge Pyramid, the DIKW Pyramid [53], is introduced as a way to represent the structural and functional relationships between Data, Information, Knowledge and Wisdom [52].

- The Data level includes all the raw-data which are collected from things, through their IoT services

- The Information level includes all the information produced by analyzing the raw data. In the Information level, things are considered situational aware.

- The Knowledge level includes problems or detected situations associate with specific solutions (cases). In this level, things acquire the power of learning from previous experiences. This is achieved with the use of a Knowledge Base (KB), which can be shared between things with suitable social characteristics. This base is used to improve the decision making and lead to an optimal solution. It is also possible for things without a KB to take advantage of the KB of their social group.

- The Wisdom level includes high-level reasoning techniques, such as Case Based Reasoning (CBR). This allows things to reason and understand their situation and make independent decisions. In this level, things are characterized as cognitive, intelligent or Wise, because they now have the ability to acquire, adapt, modify, extend and use knowledge in order to solve problems.

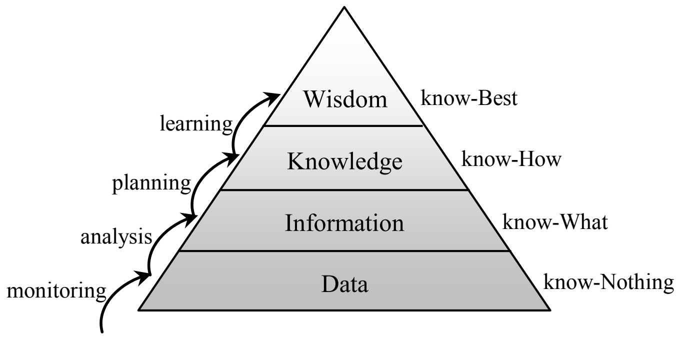


***Figure 2.4.3**: The COSMOS DIKW Pyramid* [52]

# Chapter 3: Defining the Main Ontologies

## 3.1 Introduction

The COSMOS project defines three main ontologies for the social Internet of Things:

a) Virtual Entities
b) Applications
c) Users

This chapter focuses on providing a basic understanding for each ontology and its necessity to the social Internet of Things, as well as defining the basic properties needed to achieve proper function and interoperability.

## 3.2 Virtual Entities

### 3.2.1 Introduction

Undeniably, the main component of the social Internet of Things are the Things; the objects that are able to communicate with each other and form a social network, in order to exchange information. In order to create this network, a transition from the real world to the virtual world is needed. The COSMOS project represents Things and groups of Things of the real world via their virtual counterparts: Virtual Entities (VEs). VEs may have their own goals and be equipped with an internal logic in order to achieve them. They acquire perception through accessing sensor readings via IoT-services and can impact their environment or undertake physical actions using actuators via other IoT-services. Finally, VEs may interact with each other for various purposes such as collaboration, cooperation etc. [51]. VEs are defined by their basic properties. These properties are used to characterize the VE, by providing enough information to achieve a full description of the real world object to each virtual entity. It is possible for users to add user-based properties to their objects' VEs, allowing for further customization.

### 3.2.2 VE Properties

➢ *Domain*

The Domain property is responsible for matching the VE to a given sIoT Domain. It is the first step into identifying the different groups into which VEs are organized. The Domain Property plays an important role in various Object Relationships. It is the main property required for the *Domain Object Relationship* and is also used in the *Conflict of Interest Object Relationship* to determine and resolve conflicts among VEs. The COSMOS platform should give the developer the option to choose from a variety of diverse domains. As mentioned in the specific chapter, possible choices include Traffic Congestion, Air Pollution, River Floods, Smart Grid, Meteorological Station Network, Radiation Levels, Animal Tracking, Supply Chain Control, Intrusion Detection Systems and Patients Surveillance. These domains could also provide information for further social analysis. VEs must be associated with at least one domain. Developers will have access to the full list of available domains and may be able to make suggestions of missing domains as well. By dividing our ontology's scope into domain-specific parts we also achieve a functioning segregation of available IoT-services [51]. This means that, if a certain VE exposes services with a multitude of purposes, intra-VE communication will be more effective as far as both discovery and service recognition are concerned. Therefore, if a certain VE desires a look-up of services pertaining to traffic management, domain identifiers can be used to limit the time needed for a query response.

➢ *Physical Entity*

The Physical Entity property indicates the type of the actual physical entity represented by the VE. Although relations with different types of Physical Entities are not limited by the platform, VEs that share the same Physical Entity should also be able to communicate and form relationships easier than unrelated ones. Accepted values of the Physical Entity property must be clear and indicative of the actual nature of the object to avoid confusion. Typical Values include "BusStop", "TrafficLight", "Bus", "House" etc.

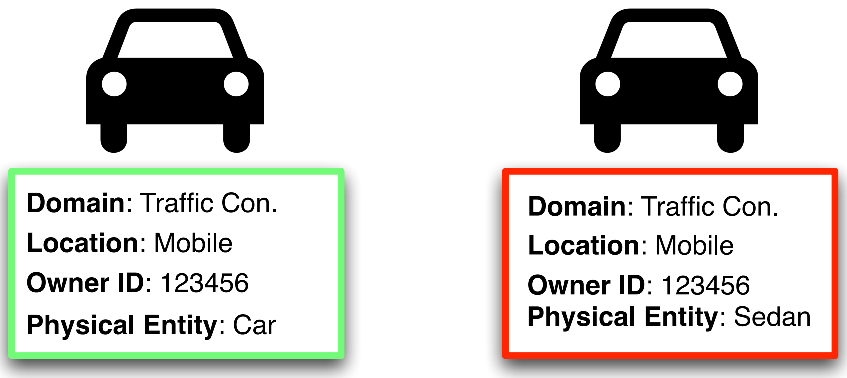


***Figure 3.2.2.1**: Acceptable and non acceptable Physical Entities examples*

➢ *Location*

The Location property is used to identify the nature of the object with regards to location setup. This property is used to distinguish moving objects from stationary ones and is important in order to determine the capacity of the VE in terms of accessibility. The location property can take the following values [51]:

- **Fixed**: This value is used for entities that are established in a permanent structure and not intended for portable operation. Examples of fixed entities include buildings such as houses, offices and hospitals

- **Portable**: This value is used for entities that are fitted in a temporary location. These entities do not have a permanent structure but may be stationary for long periods of time as well. Examples of portable entities include devices such as laptops.

- **Mobile**: This value is used for entities that can move and can therefore change their position frequently and continuously. These types of entities do not remain stationary for long periods of time but are able to relocate numerous times in a short amount of time. A prime example of mobile entities include mobile phones.

➢ *Geo-location*

The Geo-location property is the property that matches a virtual entity to the location of its physical entity. This property contains the necessary coordinates to determine the location of the VE on the map, in terms of Latitude and Longitude. Geo-location variables are represented in the ontology through the use of the "hasGeoLat" and "hasGeoLon" data-type properties that use a range of float numbers to accurately store latitude and longitude respectively. The Geo-location property is heavily used in location based relationships such as Co - Location Object Relationship, where proximity of objects is determined by their relative positions with regards to predefined geographic boundaries. Accuracy of the geo-location property is desirable, depending on the accuracy levels needed by applications to establish location based relationships. For some large scale applications, accuracy is not a priority and will not greatly affect the validity of the formed relationships. However, some applications rely on detail, which means that the smallest error on accuracy can lead to a completely different network of location based relationships.

➢ *Dependability Indexes*

The Dependability Indexes are used to determine if a VE is dependable. VEs seek information from highly dependable VEs that can be reliable, trustworthy and reputable. The social ontology contains three social indexes for this purpose[51]:

- **Reliability Index**: An absolute indicator of the performance of the physical entity that quantifies the efficiency of its sensors and actuators functionalities, relative to their normal operation. The index is represented by the data-type property "ReliabilityIndex" which contains a float from 0 to 1.

- **Trust Index**: A counter which states how many times a VE has successfully shared its CB and/or IoT-services. Coupled with the concept of feedback and through refinement of its calculation, we can use this index as a means to simulate social mobility in the platform, as Trust will be one of the most important components of friendship recommendation. The index is represented by the data-type property "TrustIndex" which contains an integer.

- **Reputation Index**: A counter which monitors how many times the VE has received a request (how many "hits" it has). It is a cumulative and comparative indicator. The index is represented by the data-type property "ReputationIndex" which contains an integer.

➢ *Owner ID*

The Owner ID property represents the physical owner of a VE. This property is used to match a VE to a user that is considered its owner. A physical owner can be either an individual or a group of individuals, such as an organization. The Owner ID is important in establishing user based relationships, such as the Ownership Object Relationship.

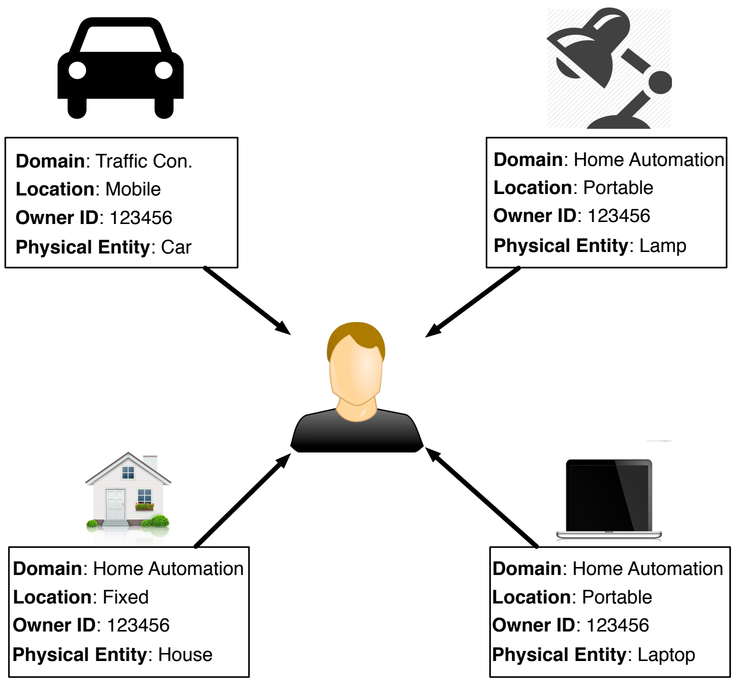


***Figure* 3.2.2.2**: *VEs owned by the same user*

➢ *VE ID*

A unique Identifier of each Virtual Entity. VE ID can be a sequence of letters and numbers, uniquely assigned to each VE once they are first registered with the platform. This property is needed to identify VEs for direct VE communication and VE discovery purposes. The VE ID is used when adding a VE to certain types of VE Lists, such as the Followees and Enemies Lists, or an application's List of VEs.

➢ *Simple Users List*

The Simple Users List property is a list that contains a number of User IDs. These User IDs represent the Simple Users that are currently using the object. This list is used to match the VE to its users. The Simple Users List is necessary in establishing user based relationships such as the Usage/Interaction Object Relationship.

➢ *Followees List*

The Followees List property is a list that contains a number of VE IDs. These VE IDs represent the VEs that are currently tracked by the VE. This list is used to match the VE to other VEs that have been determined to be dependable. Following another VE is a way of establishing friendship between the two VEs. Following a VE can be made either by user preference, when the user decides to manually add a certain VE to the Followees List, or by recommendation from the platform. Recommendations take into consideration existing followees as well as the three Dependability Indexes (Reliability Index, Trust Index and Reputation Index) in order to recommend similar VEs. The Followees List is necessary in establishing VE based relationships such as the Followers/Followees Object Relationship.

➢ *Enemies List*

The Enemies List property is a list that contains a number of VE IDs that represent the VEs that are currently considered hostile by the VE. This list is used to keep track of VEs that have been determined to be untrustworthy. Marking another VE as an enemy can be made either by user preference, when the user decides to manually add a certain VE to the Enemies List, or by recommendation from the platform. Recommendations take into consideration existing enemies as well as the three Dependability Indexes (Reliability Index, Trust Index and Reputation Index) in order to recommend similar VEs with low indexes. The Enemies List is necessary in establishing VE based relationships such as the Enemy Object Relationship.

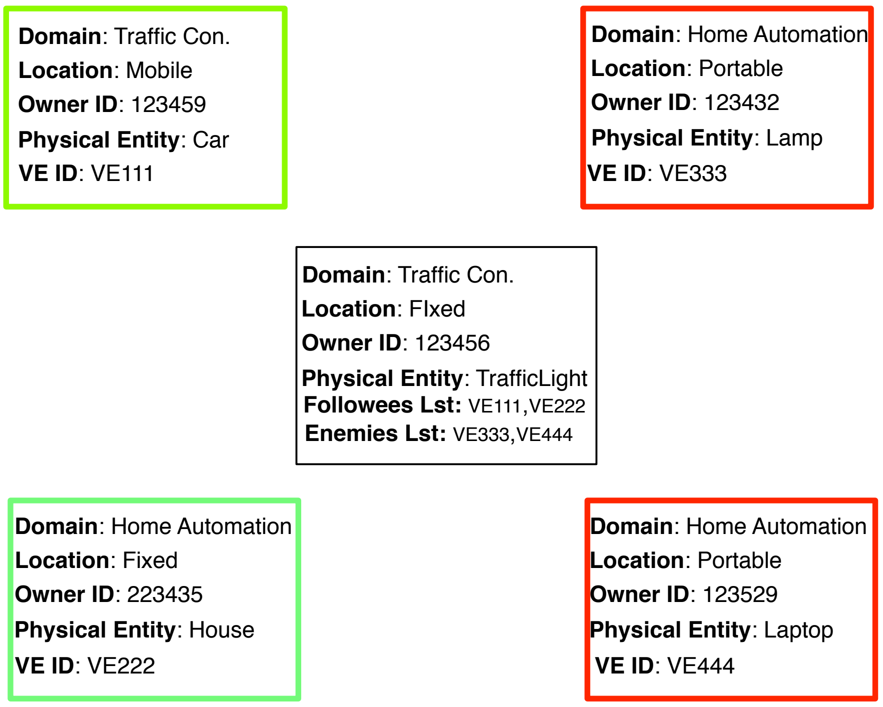


***Figure* 3.2.2.3**: *Enemies and Followees of a VE*

➢ *List of Applications*

The List of Applications is a list containing all the applications that the VE is currently associated with. A VE can be associated with more than one applications at a time, depending on resources, availability and application needs. The List of Applications can be used to identify groups of VEs, via their application associations. VEs working in the same application must be able to communicate easier than unrelated objects in different applications. The List of Applications can be used in establishing VE based relationships such as the Co - Work Object Relationship or the Parental Object Relationship, as well as in application based Object Relationships such as the Replacement Object Relationship.

➢ *List of IoT Services*

The List of IoT Services is a list containing all the IoT services that the VE can offer. A VE can offer a variety of IoT services depending on its description and nature. The List of IoT Services is important in informing other VEs in search of these services that this VE does indeed support them. IoT Services play an important part in the knowledge flow in the social Internet of Things, as well as in the sharing of services among VEs. The List of IoT Services can be used in establishing application based relationships such as the Replacement Object Relationship.

➢ *Financial Packages*

The Financial Packages property is used to match a VE with its financial information, regarding applications that offer a payment model in the form of financial packages. Financial packages are used to certify if and what type of benefits a VE has been awarded by the application administration. Examples of benefits include extended cloud storage space, additional resources, access to limited material or priority in service. The Financial Packages property is a list containing pairs of applications and the financial package the VE has acquired for the corresponding application. Possible values of Financial Packages are determined by the application and will be discussed in the following sections.

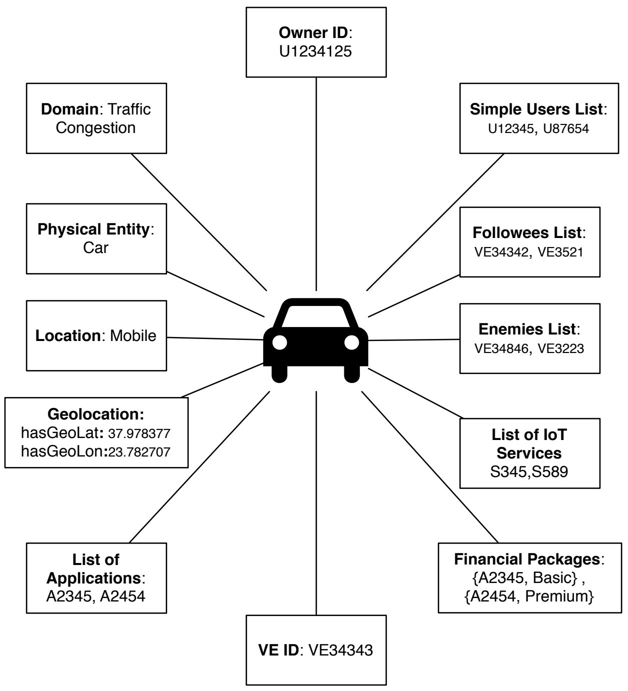

Owner ID: U1234125
Domain: Traffic Congestion
Simple Users List: U12345, U87654
Physical Entity: Car
Followees List: VE34342, VE3521
Location: Mobile
Enemies List: VE34846, VE3223
Geolocation: hasGeoLat: 37.978377 hasGeoLon:23.782707
List of IoT Services S345,S589
List of Applications: A2345, A2454
Financial Packages: {A2345, Basic}, {A2454, Premium}
VE ID: VE34343

***Figure 3.2.2.4***: *VE Properties*

## 3.3 Applications

### 3.3.1 Introduction

Another significant ontology in the social Internet of Things regards Applications. An application can be defined as a set of tasks, activities or goals that take place in the social Internet of Things. Applications are created by Users and are carried out by Virtual Entities. Each VE is assigned to a number of applications in the sIoT and is responsible for fulfilling a number of tasks and activities for those applications. A notable example of an application is a City Waste and Recycling Management Application, which consists of tasks for cleaning the city and taking care of recycling. Examples of VEs participating in such an application include garbage disposal trucks or recycling dumpsters. These VEs are responsible for carrying out the necessary tasks for achieving the goal of the application, such as collecting garbage from dumpsters, or reporting on the capacity of each dumpster.

Applications are defined by their basic properties. These properties are used to provide enough information to achieve a full description of the application and its goals. It is possible for users to add user-based properties to their applications, allowing for further customization.

### 3.3.2 Application Properties

➢ *Application ID*

A unique Identifier of each application. Application ID can be a sequence of letters and numbers, uniquely assigned to each application once they are first created in the social internet of things. This property is needed to identify applications as well as aid in application discovery. The *List of Applications* VE property contains a list of Application IDs that match the VE to the applications it has joined.

➢ *Application Info*

The application Info property is introduced to identify basic information regarding the application. This information is used to shape the application's purpose in the social internet of things and provide useful information to users aiming to register their VEs to the application. The Application Info property consists of the following:

- **Application Description**: A brief description of the application. Application description is set by the creator of the application and is used to inform other entities in the network of the application's identity. This description must be brief and cochise, to avoid user misinformation.

- **Application Duration**: The type of the time duration of the Application. Duration of an application can be a fixed period of time, in which case this property contains the value of the duration formatted properly to display years, days, hours, minutes, seconds. Duration of an application can also be open, in which case no time limit has been assigned to the application and the value of the property is set to *Open*.

- **Application Start Date**: The date that the application was initialized. The creator of an application is responsible for the initialization of the application. All applications must have a valid Application Start Date upon their creation.

- **Application Expiration Date**: The date that the application will be terminated. This property is dependent on the Application Duration property. If the application has been configured with a fixed period of time in mind, then this property contains the date that the application will come to an end. On the contrary, applications configured with an *Open* application duration do not have an expiration date. This property is set to *Open* instead, in this case.

- **Application Size**: The size of the application. Size is measured in the number of VEs that is involved with the application. This property can be used as a measurement of magnitude among application, as well as statistics and grouping purposes.

- **Application Goals**: Applications are usually created by their developers with certain goals in mind. This property is used to document those goals and monitor the application progress, in terms of accomplished goals, throughout the duration of the application. Application goals are set by the creator of the application. Automating monitoring of goal completion can be a challenge, as there is no way of communicating this quantity. Developers must be responsible for monitoring goals and determining if a goal is completed or not.

➢ *List of VEs*

This property is a list containing all the VEs that are currently associated with the application. This list is dynamic, as VEs join and leave applications constantly. The List of VEs contains the VE IDs of these entities. This property is necessary in establishing application based relationships such as the Co - Work Object Relationship as well as VE based relationships such as the Parental Object Relationship.

➢ *Head VE*

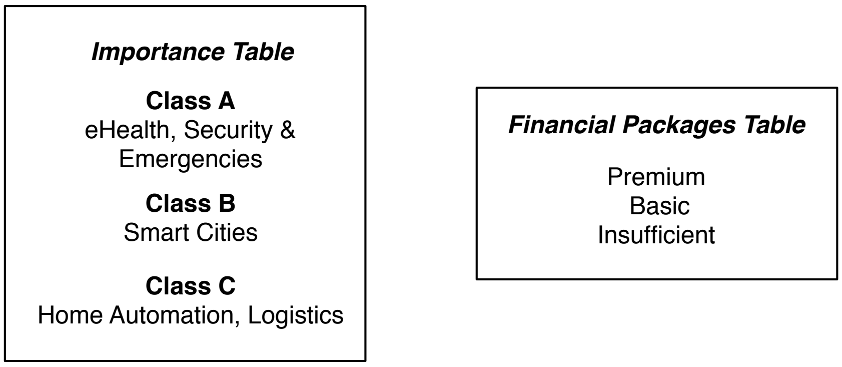
The Head VE property is used to identify a Virtual Entity that has elevated authority over the rest of the VEs of the application. This authority allows the Head VE to control and guide the group of VEs, when needed. This property must be configured by the creator of the application and contains the VE ID of the chosen Head VE. The Head VE is necessary in establishing the VE based Parental Object Relationship.

➢ *Importance Table*

This property is a table that contains all the available domains sorted based on priority. Priority is established by the developer of the application during its creation. Developers are given the list of available domains and are responsible for categorizing these domains in priority classes. The number and the size of classes that are created are entirely up to the developer as well. The importance table is needed to introduce the element of hierarchy among domains. This hierarchy plays an important role in relationships where resource sharing creates deadlocks among objects aiming for limited resources. The Importance Table is used to resolve conflicts as witnessed in the application based Conflict of Interest Object Relationship.

➢ *Financial Packages Table*

This property is a table that contains all the available financial packages, sorted based on priority. Financial packages are used to introduce the use of economic models by applications. Some applications offer a choice of free or paid packages that can offer various types of  enhancements, such as access to limited content or priority in shared resources. Financial packages are configured by the developer of the application who is also responsible for confirming the purchase of these packages by VEs. This property plays an important role in relationships where resource sharing creates deadlocks among objects aiming for limited resources. The Financial Packages Table is used to resolve conflicts as witnessed in the application based Conflict of Interest Object Relationship.


***Importance Table***

**Class A**
eHealth, Security & Emergencies

**Class B**
Smart Cities

**Class C**
Home Automation, Logistics

***Financial Packages Table***

Premium
Basic
Insufficient

***Figure 3.3.2**: Example of Importance and Financial Packages Tables*

➢ *GeoBoundaries*

This property is used to identify the geographical boundaries inside which the application is operated and is considered valid. The GeoBoundaries is a list of coordinate pairs that are used to map a certain area. The creator of the application is responsible for the correct input of those geographical boundaries. GeoBoundaries play an important role in location based relationships. The Co - Location Object Relationship relies heavily on GeoBoundaries to determine proximity among objects of an application.

➢ *Creator ID*

The Creator ID contains the User ID of the user responsible for creating the application. This property is used to match users to their applications and can be used for statistics and grouping purposes.

## 3.4 Users

### 3.4.1 Introduction

Autonomy among objects can only be established when we eliminate the need of human involvement in the Internet of Things. Objects are able to acquire knowledge and become "wise", by having the ability to use the acquired knowledge to make decisions by themselves. A basic component, therefore, for achieving autonomy, is the knowledge flow that is created through the social Internet of Things. The goal of the sIoT is to enable communication between objects and not users.

It is understood that in the Internet of Things, the desired role of human users is non existent. This, however, cannot be true, as with all systems, the presence of users is, to some extent, necessary for various reasons. The presence of special kinds of users called network administrators is common practice in computer networks. These users are responsible for maintaining the network by monitoring traffic. Security is also another aspect that needs user interference, with moderators needed to inspect suspicious traffic and decide if content violates rules of conduct.

Human users also play a part in the formation of some of the proposed relationships. Human interaction with objects is needed in the calculation of relationships regarding ownership and usage of objects. Users need to report which objects are being owned or currently used by them, as objects are not able to identify owners and users by themselves. Users are also responsible with associating their objects with the available domains and applications that are offered in the social Internet of Things.

Another aspect of human involvement that can be found in the sIoT is actual relationships that users have in their human network. Although it has been stated that social relationships need to be established among things and not as an outcome of the social network of humans, users have the option to involve some of their social network elements to the social Internet of Things. Through the interaction with other people, users develop social relationships in the form of friends and enemies. When someone labels a person as a friend, then it is understood that a level of trust is established. People tend to seek advice from their friends, as well as other notable figures that they consider trustworthy. In the same context, an enemy is a person that cannot be trusted. Users have the ability to translate these two relationships in the object space, by being involved in the formation of similar object relationships.

### 3.4.2 Types of Users

We can identify three types of users in the social internet of things. The first type of user is called **administrator**. An administrator is responsible for matters concerning maintenance, security and observation. In order to achieve this, the administrator is given a high level of permissions and privileges, as opposed to other types of users. However, administrator power is properly distributed to ensure information and user data privacy. Two other types of users are called **owner** and **simple user**. As the name suggests, an owner is a user that owns one or more objects that can be found in the social Internet of Things. A simple user, on the other hand, is any user that interacts with one or more objects that can be found in the sIoT. The main difference between owners and simple users is that owners have more power over the objects they own, in terms of permissions. Characterization of user type is not exclusive. An owner of an object can also be a simple user of other objects and vice versa. All types of users have the ability to create new applications, define their characteristics and properties and add, invite or expel VEs from the application they created.

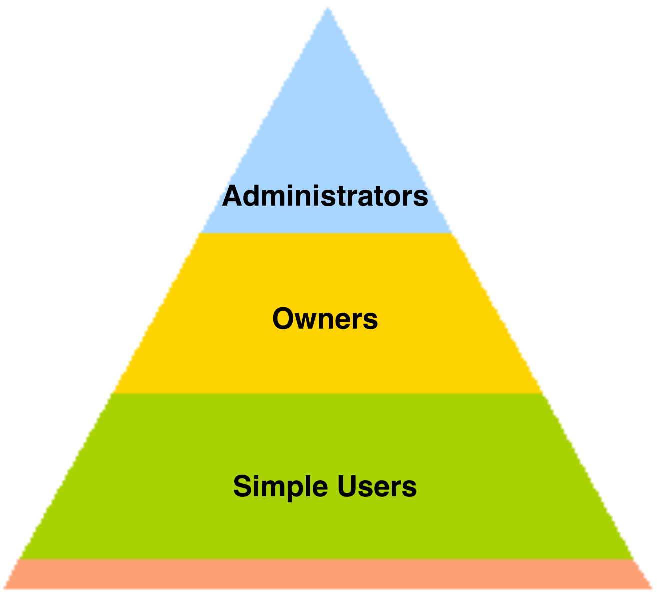


***Figure 3.4.2***: *Hierarchy of types of users*

### 3.4.3 User Properties

The properties of each user are the following:

**User ID**: A unique user Identifier of each user. User ID can be a sequence of letters and numbers, uniquely assigned to each user once they first interact with the social Internet of Things. This property is needed to identify users as well as offer a level of privacy, as the full name is never involved in the formation of relationships. The User ID is important for the ownership and usage/interaction relationships, as well as the followers and enemies relationships.

**Full Name**: The full name of the user. This information is used to match a User ID to a user name and facilitate the search of friends or enemies. As it is sensitive information, it is never involved in the knowledge flow. User ID is involved instead.

**Email**: The email of the user. This information is used for registration, identity verification and communication purposes.

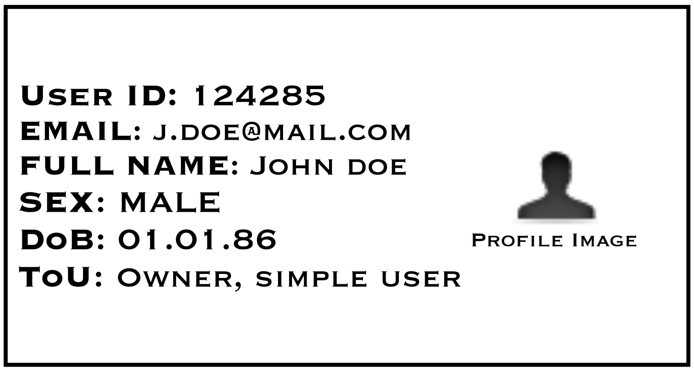
**Sex**: The sex of the user, i.e male or female. Another type of identifier. Can be used for statistics or grouping purposes.

**Date of Birth(DoB)**: The date of birth of a user. Can be used for statistics, grouping purposes or age restriction of services.

**Profile Image**: An image of the user. This is mainly used for identification purposes during the search of friends or enemies.

**Types of User(ToU)**: The type or types that the user has on the social Internet of Things. A user can either be an administrator, an owner, a simple user, or a combination. Can be used for statistics or grouping purposes.

Of the above properties, only the User ID is actively involved in the formation of relationships. The basic user info (full name, email sex, DoB, profile image and ToU) offer supplementary information about the user and are optional.



USER ID: 124285
EMAIL: J.DOE@MAIL.COM
FULL NAME: JOHN DOE
SEX: MALE
DOB: 01.01.86
TOU: OWNER, SIMPLE USER

PROFILE IMAGE

***Figure 3.4.3***: *User Properties*

# Chapter 4: Relational Models and Hierarchy

## 4.1 Relational Models

### 4.1.1 Introduction

This chapter is dedicated to presenting the proposed Relational Models that will serve the social aspect in the social Internet of Things. These are:

- Ownership Object Relationship
- Usage/Interaction Object Relationship
- Domain Object Relationship
- Followers/Followees Object Relationship
- Enemies Object Relationship
- Parental Object Relationship
- Co - Work Object Relationship
- Conflict Object Relationship
- Replacement Object Relationship
- Co - Location Object Relationship

For each of the above Object Relationships, information regarding characteristics, requirements for formation, RMT elements, challenges and motivation are provided, along with detailed examples of scenarios.

### 4.1.2 Ownership Object Relationship

The Ownership Object Relationship is a simple relationship which is established between objects that share the same owner. The owner of an object is defined as the user who has full access to the object, regarding its use, maintenance and user permissions. Although many users can have various levels of access to the referred object, only the owner can at any point have full control of the object. Typical examples of owners include the administrator user of a computer or the owner of an automobile. In these examples, although other users can use the objects, in this case the computer, or the car, some operations are reserved only for the owner. Ownership is typically established at the time of purchase of the object, where certain paperwork certifies the owner. A high level of security is offered to the owner, in the form of passwords or car keys, for the above examples.

The Ownership Object Relationship is primarily a **user based** relationship. This means that the users play an important role in the formation of these kinds of relationships. The less frequent change of owners of an object makes the relationship a more **static** relationship. This means that ownership relationships are not subject to frequent changes and have a longer duration than other types of relationships. This duration also makes the relationship more **reliable** than others, as there is less room for errors during its calculation. The Ownership Object Relationship is a **symmetric** relationship, meaning that if an object has an ownership relationship with another object, then the same applies for the second object. In terms of Relational Models Theory, the Ownership Object

Relationship integrates two of the four basic relational structures, **Equality Matching** and **Authority Ranking** [54] .

The main property that is needed to establish an ownership relationship between objects is the **Owner ID**. The Owner ID is an identification that matches the object to the user that owns it and is assigned to the Virtual Entity of the object at the beginning of ownership. This property can be changed only if and when the object changes owners. Calculating the relationship is fairly trivial; if two Virtual Entities share the same Owner ID, then they form an ownership relationship. The matching between objects and owners is done with the use of the User ID. The User ID of the user who owns an object is assigned to the Owner ID of the object.

The motivation behind this relationship lies in the need of knowledge flow between objects that share the same owner. Through interaction with the owner and with the use of sensors, objects are able to acquire information that could prove useful to the other objects that are owned by a user. Missing information is a common problem for objects that are fairly "young" and haven't yet acquired a lot of information on their own. Through the ownership relationship, however, these objects can accelerate the process of achieving autonomy by filling the missing information through querying other objects.

A minor challenge regarding this relationship regards the need of user involvement. The user needs to be actively involved with the object, in order be registered as its owner. Up to this point there has not been any way for objects to detect their owners by themselves, without the interference of humans. However, this is a fairly trivial challenge, as a well designed interface can facilitate this procedure of registration. Moreover, the user involvement begins and ends at the registration level, which makes user involvement not essential for objects to achieve autonomy in the network. After the user has registered his/her objects, this relationship needs no more information from the user, in order to be established.

A simple IoT scenario regarding the Ownership Object Relationship is the following. John and Mary are working for the same company and have adjacent offices on the same floor. Both John and Mary have recently bought various IoT enabled lamps, which are equipped with sensors to measure the level of brightness in the vicinity. These lamps can offer an IoT service in the form of adjusting their brightness to their users' needs. Both of them have installed these lamps at their houses. Through the sensors and the frequent adjustment by their users, these lamps have "learned" to adjust their brightness based on their owners preferences and the brightness of the room they are in. John decides to buy one more lamp for his office. Once installed, the office lamps establishes an ownership relationship with John's home lamp and is quick to adjust its brightness according to John's preferences. Once Mary decides to install the same type of lamp in her office, this lamp will establish an ownership relationship only with Mary's home lamp, as only these lamps share the same Owner ID.

The above scenario is displayed in the figure below, where VE1 and VE4 are John's home lamp an office lamp respectively, and VE2 and VE3 are Mary's home and office lamps.

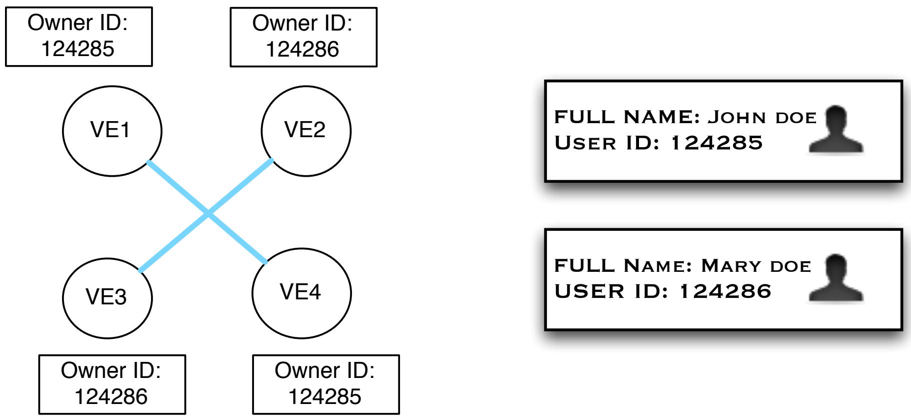


***Figure 4.1.2**: Ownership Object Relationship scenario*

### 4.1.3 Usage/Interaction Object Relationship

The Usage/Interaction Object Relationship is similar to the ownership relationship, with a notable difference in the types of users that take part in its creation. This relationship is established between objects that share a common simple user. A simple user of an object is defined as the user who has partial access to the object, regarding its use, maintenance and user permissions. Simple users can have various levels of access to the referred object, however the extent of this access is decided only by the owner, who has full control of the object. Typical examples of simple users include a guest user of a mobile phone or the user of car rental service. In these examples, although simple users do not own these objects, in this case the phone, or the car, some operations are made available to them by the owner. Usage is typically established in the form of an agreement with the owner or by registering and logging in to a service . A high level of security is offered to the simple users as well.

The Usage/Interaction Object Relationship is primarily a **user based** relationship. This means that the users play an important role in the formation of these kinds of relationships. The frequent change of active users of an object makes the relationship a more **dynamic** relationship. This means that these relationships are subject to frequent changes and have a shorter duration than other types of relationships. However, this relationship is more **reliable** than others, as there is less room for errors during its calculation, a trait shared with the ownership relationship. The usage/interaction relationship is a **symmetric** relationship, meaning that if an object has a usage/interaction relationship with another object, then the same applies for the second object. In terms of Relational Models Theory, this relationship integrates two of the four basic relational structures, **Equality Matching** and **Authority Ranking**.

The main property that is needed to establish a usage/interaction relationship between objects is the **Simple Users List**. The Simple Users List is a list containing all the User IDs of simple users that are actively using the object. This property can be changed in two ways; when new simple users acquire access to the object, in which case their ID is added to the list, or when a user opts out of using the object, in which case their ID is deleted from the list. Calculating the relationship is fairly trivial; if two Virtual Entities share a simple user, then they form a usage/interaction relationship.

The motivation behind this relationship lies in the need of knowledge flow between objects that share simple users. Through interaction with these users and with the use of sensors, objects are able to acquire information that could prove useful to the other objects that are used by the same simple user. Missing information is a common problem for objects that are fairly "young" and haven't yet acquired a lot of information on their own. Through this relationship, however, these objects can accelerate the process of achieving autonomy by filling the missing information through querying other objects. In contrast to the ownership relationship, the usage/relationship relationship is a more flexible relationship, because of the frequency with which users alternate between objects as well as the volume of objects a simple user interacts with.

A minor challenge regarding this relationship regards the need of user involvement. This has been previously presented for the ownership relationship as well.The user needs to be actively involved with the object, in order be registered as its simple user. Up to this point there has not been any way for objects to detect their simple users by themselves, without the interference of humans. However, this is a fairly trivial challenge, as a well designed interface can facilitate this procedure, much like the registration for the ownership relationship. Moreover, the user involvement begins and ends at this level, which makes user involvement not essential for objects to achieve autonomy in the network. After the user has been declared an active user for his/her objects, this relationship needs no more information from the user, in order to be established.

A simple IoT scenario regarding this relationship is the following. Bob lives in Greece and drives an IoT enabled vehicle. This vehicle is equipped with sensors that gather information regarding Bob's driving habits, as well as various other types of information. Bob is required by his employer to travel to Italy for six months. On his first day in the country, he rents a new car, IoT enabled as well. By logging in, the new car is able to establish a usage/interaction relationship with the first car and get information regarding Bob's driving habits from the other car. When Bob is required to go to France for a few days, another rented car can follow the same process as the previous one to instantly get the information from the previous vehicles that Bob has used.

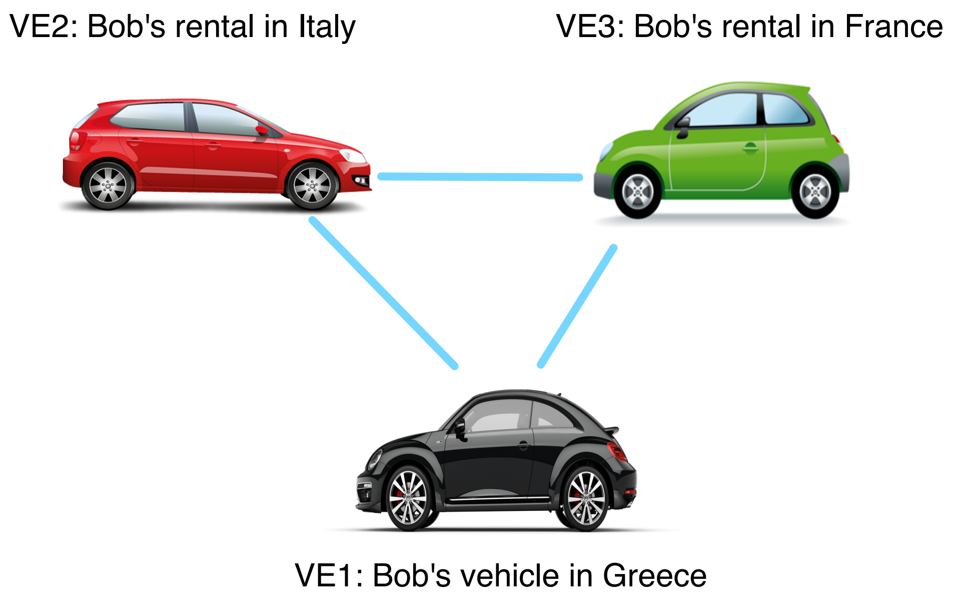

VE2: Bob's rental in Italy
VE3: Bob's rental in France
VE1: Bob's vehicle in Greece

***Figure 4.1.3**: Usage/Interaction Object Relationship scenario*

### 4.1.4 Domain Object Relationship

The Domain Object Relationship is another basic relationship that is established between objects that share the same domain. As detailed in previous chapters, domains are separated into the following main categories:

• Smart Cities
• Smart Environment
• Smart Metering
• Security & Emergencies
• Retail
• Logistics
• Home Automation
• eHealth
• Smart Animal Farming
• Smart Water
• Smart Agriculture

These categories, along with their numerous sub-categories, offer a diverse selection for object identification. Examples of objects in the "Waste Management" domain, for example, include smart garbage collection vehicles and smart garbage bins. Domain characterization is usually established by owners or users of the object, who are given a list of domain options to choose from when they set up their objects. The list of domains can be extended with new domain suggestions from users.

The domain relationship is primarily a **user based** relationship. This means that the users play an important role in the formation of these kinds of relationships. Domains are usually declared at the very beginning. The less frequent change of domains of an object makes the relationship a more **static** relationship. This means that domain relationships are not subject to frequent changes and have a longer duration than other types of relationships. This duration also makes the relationship more **reliable** than others, as there is less room for errors during its calculation. The domain relationship is a **symmetric** relationship, meaning that if an object has a domain relationship with another object, then the same applies for the second object. In terms of Relational Models Theory, the domain relationship integrates two of the four basic relational structures, **Equality Matching** and **Communal Sharing**.

The main property that is needed to establish a domain relationship between objects is the **Domain** property. The Domain property is an identification that matches the object to a certain predefined domain. This property can be changed only if and when the object changes domains. Calculating the relationship is fairly trivial; if two Virtual Entities share the same Domain, then they form a domain relationship.

Minor challenges regarding the domain relationship include the involvement of users. Users are needed to assign a certain domain to the object. However, as with the previous two relationships, this involvement is minimum. Another minor challenge with this relationship regards the choice of domains. Although a plethora of options is already available, efforts need to be made to ensure that these options are always up to date. The ability to allow users to add their own domains also requires some effort to maintain the list as cochise and small as possible, in order to avoid multiple appearances of similar domains.

The motivation behind this relationship lies in the need of knowledge flow between objects that belong to the same domain. Objects are able to acquire information that could prove useful to other objects of the same domain. Missing information is a common problem for objects that are fairly "young" and haven't yet acquired a lot of information on their own. Through this relationship, however, these objects can accelerate the process of achieving autonomy by filling the missing information through querying other objects. In some critical cases, such as eHealth, it is vital for some objects to seek information only from objects of the same domain, to reduce misinformation from similar objects that are registered under a different domain.

A simple IoT scenario regarding the domain relationship is the following: Hospitals A and B have installed a smart heating and ventilation system to better monitor and control temperature and air in rooms containing patients recovering. All patient rooms are registered under the domain of Patients Surveillance. Bob has been recovering in room 642 of Hospital A for a few days and now needs to be transferred to Hospital B to continue treatment to a newly built ward. Once Bob occupies room 373 in Hospital B, the heat system that is registered under the same domain of the previous room is able to communicate with room 642 of Hospital A to adjust the temperature to accommodate the patient. Room 731 of Hospital A is not occupied by any patient, does not have the same domain as the other two and, therefore, will not adjust the room temperature.

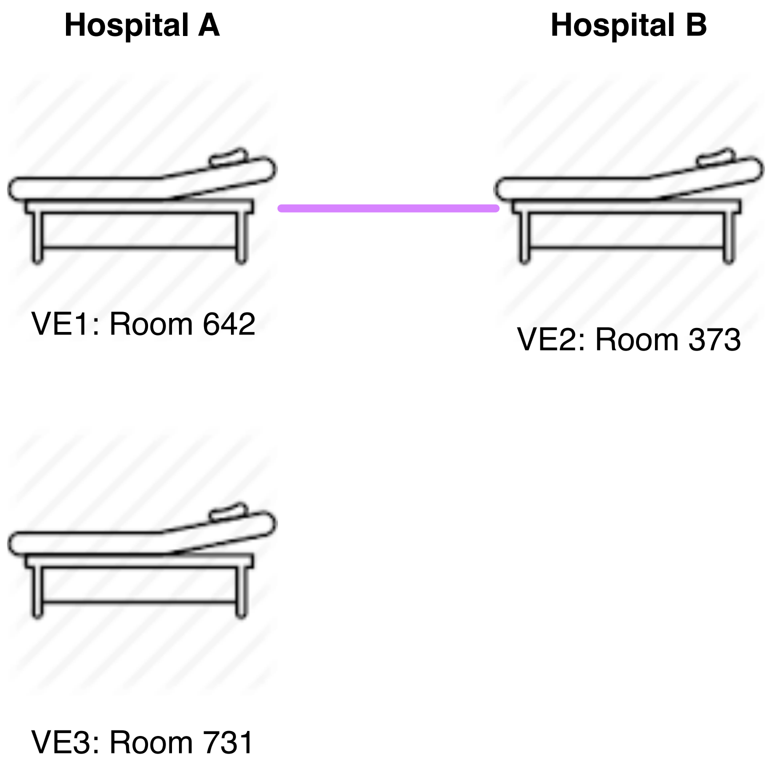


***Figure 4.1.4**: Domain Object Relationship scenario*

### 4.1.5 Followers/Followees Object Relationship

Borrowing elements from similar relationships in the social networks, the followers/ followees object relationship is established between objects that follow other objects. Following objects is similar to following other users on social networks; Objects establish a form of friendship by following objects that are considered trustworthy.

Contrary to the previous relationships, the followers/followees relationship is a **VE based** relationship. This means that Virtual Entities play an important role in the formation of these kinds of relationships. Users do not have as much involvement in these types of relationships. The ephemeral nature of forming friendships makes the relationship a more **dynamic** relationship. This means that these relationships are subject to frequent changes with various ranges of duration. Therefore, this relationship is less **reliable**. The followers/followees relationship is an **asymmetric** relationship, meaning that if an object has a follower/followee relationship with another object, then the same does not necessarily apply for the second object. In terms of Relational Models Theory, this relationship integrates two of the four basic relational structures, **Equality Matching** and **Communal Sharing**.

The main property that is needed to establish a domain relationship between objects is the **Followees List** property. The Followees list contains all other Virtual Entities that the object tracks. VEs that are found in the Followees List are blindly trusted by the chosen VE. Calculating the relationship is fairly trivial; A Virtual Entity forms a follower/followee relationship with every VE in its Followees List.

Populating the Followees List of each VE is the biggest challenge for this relationship. This process begins as early as at the phase of registration. The user can manually set the Followees List of the VE, by adding VEs of known friends from its social network. This is the most basic way a VE forms social bonds. Such friends will have a number of benefits during the social monitoring or discovery phases (e.g. greater priority). Recognizing the opportunity of a malicious user trying to 'overvalue' his/her own VEs and therefore create imbalances in the social network through collusion, the platform takes into account the specific social characteristics of the registering VE and adds random suitable VEs in its Followees List too[55].

Another way of acquiring Followees is through a discovery mechanism, which is based on recommendation. Discovery through recommendation is more reliable and provides protection from malicious behavior. New Followees can be recommended to a VE by its current Followees or by the Social Analysis component.

In the first case, VEs recommend their own Followees to other VEs. After the VE acquires a number of recommended Followees, their Dependability Indexes are calculated. Finally, based on the thresholds set by the user for these indexes, the Friends Management component decides whether it will accept the new recommendations or not. In the second case, the VE sends Followee recommendation requests to the SA component. Using weights for calculating the Dependability Index, a minimum acceptable limit of its value and the current Followees List as parameters, the SA calculates the Dependability of the Followees, based on the above input. If the new indexes are below the limit, the SA purges these VEs from the list, replacing them with more reliable ones, and a new Followees List is returned. Followees that have been set by users and do not have high Dependability Index anymore are not thrown away from the Followees List, but are isolated (are not used by the sharing-mechanisms), until they reach a desirable Dependability Index[55].

The motivation behind this relationship lies in the need of knowledge flow between objects that are considered trustworthy. Similar to social networks, objects follow other objects that have proven to have relevant and truthful information. Virtual Entities of these objects gain reputation and influence other objects based on this information. As humans tend to trust and listen to their friends more, the same holds true for objects in the social internet of things; their virtual entities are more likely to reach out to Virtual Entities in their Followees List for information first. The element of trust in this relationship is important in determining how reliable this relationship is.

A simple scenario regarding the Followers/Followees relationship is the following: Alice, Bob and Eve have all purchased smart refrigerators, that can monitor their inventory, their owners' preferences and make shopping arrangements based on this information, by ordering from a nearby supermarket. Alice and Bob are longtime friends and share the same taste in food, as they are both vegans. Bob is a famed vegan chef and his refrigerator has high reputation on the social internet of things. As a result, Bob's refrigerator is recommended by the system to Eve, another vegan, as well as Alice's, as they are friends. Based on the VEs in their Followees List, the refrigerators are able to establish Follower/ Followee relationships accordingly. If two more people, Rob and Steve, both carnivores, join the social internet of things with their refrigerators, they will be suggested to each other and not the vegans.

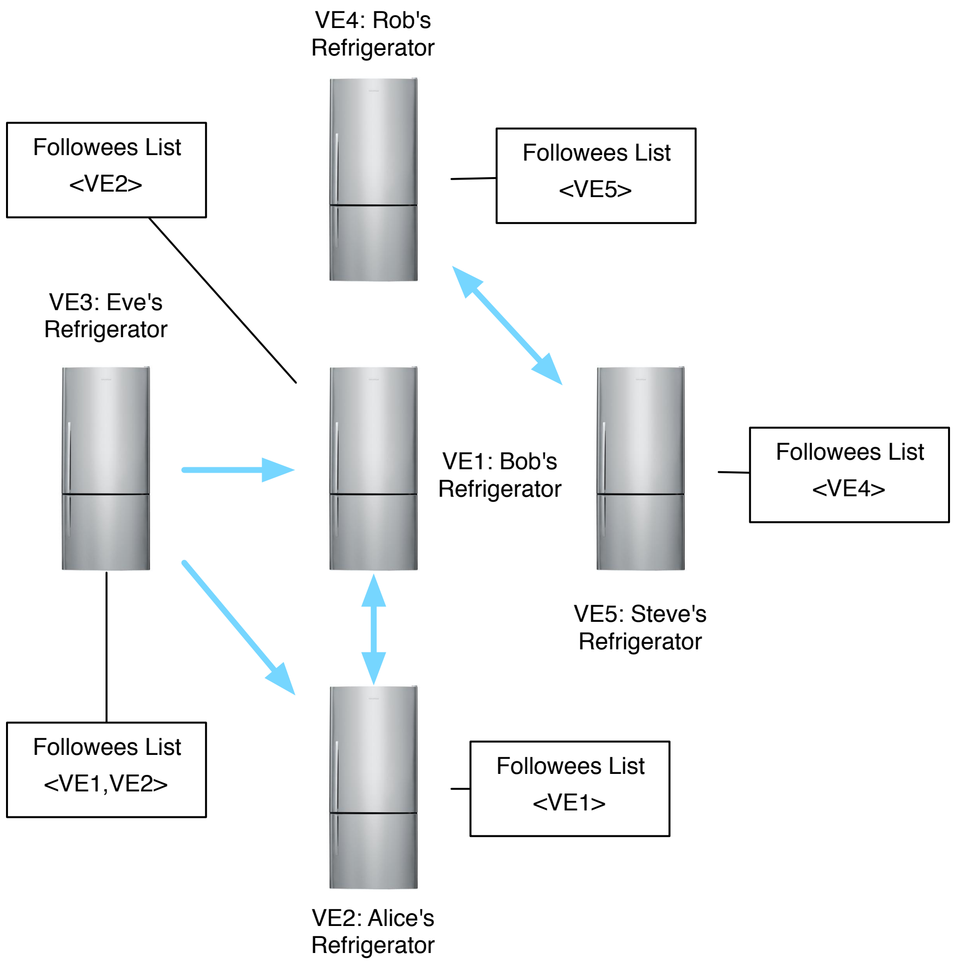


***Figure 4.1.5***: *Followers/Followees Object Relationship Scenario*

### 4.1.6 Enemy Object Relationship

The Enemy Object Relationship is the exact opposite of the Followers/Followees Relationship. This relationship is established between objects that consider other objects enemies. An object is considered hostile when it is untrustworthy. Information from untrustworthy sources should be handled with warning.

Similar to the follower/followee relationship, the enemy relationship is a **VE based** relationship. This means that Virtual Entities play an important role in the formation of these kinds of relationships. Users do not have as much involvement in these types of relationships. Making enemies is as fluid as forming friendships, which makes the relationship a more **dynamic** relationship. This means that these relationships are subject to frequent changes with various ranges of duration. Therefore, this relationship is less **reliable**. The enemy relationship is also an **asymmetric** relationship, meaning that if an object has an enemy relationship with another object, then the same does not necessarily apply for the second object. In terms of Relational Models Theory, this relationship integrates two of the four basic relational structures, **Equality Matching** and **Market Pricing**.

The main property that is needed to establish a domain relationship between objects is the **Enemies List** property. The Enemies list contains all other Virtual Entities that the object considers hostile. This list is similar to the Followees List, with the difference that any VE that is included in the Enemies List is not considered trustworthy. On the contrary, information from these VEs should be avoided. Calculating the relationship is fairly trivial; A Virtual Entity forms an enemy relationship with every VE in its Enemies List.

The Enemy Relationship shares the same challenge of populating the relevant list as the follower/followee Relationship. Similar steps with the previous relationship can be followed towards overcoming this challenge. The user can manually set the Enemies List of the VE, by adding VEs of undesired people from its social network. This is a way of transferring feelings of untrustworthiness from the social network of humans to the social internet of things. Recommendation of enemies can also come from the Social Analysis component, by following similar strategies with finding friends, by examining lower levels of indexes and adding VEs with indexes below certain thresholds as enemies.

The motivation behind this relationship is a bit different than the rest of the relationships. The goal of this relationship is not to actually maintain knowledge flow between objects, but rather to avoid flow of dubious information that can not be considered trustworthy. In order for objects to be considered autonomous, they must be able to distinguish between information they can trust and information they cannot. Continuous flow of erroneous or misleading information can jeopardize the integrity of the accumulated knowledge in the social internet of things. By identifying enemies, a VE can avoid getting information from this source, safeguarding the quality of its knowledge. If a VE is targeted as an enemy by a group of other VEs, then the network of VEs can autonomously ostracize this hostile VE. This process allows for maintaining a “healthy” community of VEs that ensure that the knowledge flow that is created is of the highest quality.

A simple scenario regarding the Enemy Object Relationship is the following: in a Group of Virtual Entities (which can be various types of objects), VE1 starts behaving in a strange way, transmitting misleading information. As a result, its Dependability indexes reach lower than the thresholds set up by other VEs, which in turn add it to their Enemies List, establishing an Enemy Relationship with the VE that started to act in an abnormal way.

The user of VE4 considers V5 untrustworthy, so it is added to its Enemies List as well. The following figure show the Enemy Relationships that are developed among the 5 VEs.

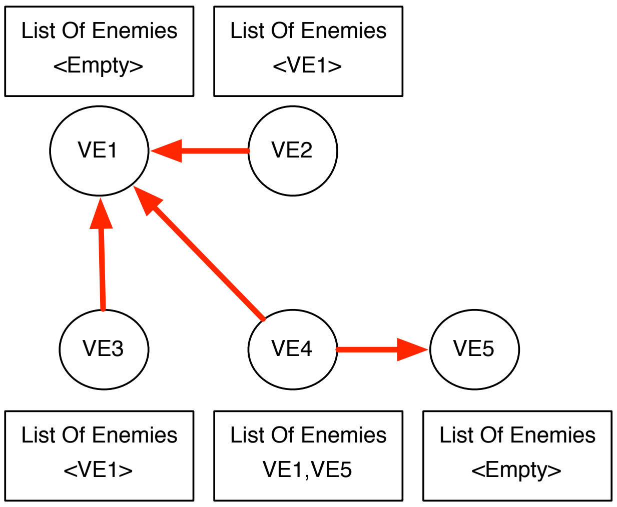


***Figure 4.1.6***: *Enemy Object Relationship scenario*

### 4.1.7 Parental Object Relationship

The Parental Object Relationship is the relationship that is established between objects and their higher ranked counterparts. A higher ranked object is defined as the object that has permission to monitor and control, to some extent, objects that are configured to be under its authority. In the human network analogy, a traffic officer has a parental relationship with all drivers approaching the intersection. This means that he/she has the right to control traffic at that certain point, by stopping cars and allowing other cars to pass as he/she sees fit. In computing,  a parental relationship exists between a scheduler and various threads, processes or data flows that are all vying for access to system resources such as CPU time and communications bandwidth. The scheduler is responsible to apply certain scheduling policies in order to distribute these resources to achieve maximum throughput, fairness and the least possible latency and waiting time. The same idea can also be applied to the social internet of things, with objects playing the role of traffic officers or computer schedulers.

The parental relationship is a **VE based** relationship. This means that Virtual Entities play an important role in the formation of these kinds of relationships. Users do not have as much involvement in these types of relationships. Authorities are decided per application needs and are not that often subject to change, which makes the relationship a more **static** relationship. This ensures a certain duration that makes the relationship more **reliable** than others. The parental relationship is also a strictly **asymmetric** relationship, meaning that if an object has a parental relationship with another object, then the same does not apply for the second object. In terms of Relational Models Theory, this relationship integrates three of the four basic relational structures, **Equality Matching** and **Market Pricing** and **Authority Ranking**. In fact, it is the most prevalent example of Authority Ranking among all relationships, as the presence of a higher authority is necessary for this relationship to exist.

The main properties that are needed to establish a parental relationship between objects is the **Head VE** property and the **List of VEs** for the application. The Head VE is the object that has authority over all the VEs in the List of VEs. Again, calculating the parental relationship is as trivial as the previous relationships so far; the Head VE establishes a parental relationship with every VE in the List of VEs.

The main challenge with establishing a parental relationship lies in the need to define a Head VE. Chain of authority requires that a higher rank must appoint a VE as a Head VE. This means that other VEs cannot in any way decide their Head VE. The Head VE needs to be predefined by the application that all VEs are used for instead. Therefore, this property is set by users creating applications for the social Internet of Things. However, as with previous user involvement, it is limited only at an initial stage of setup, thus not requiring further user presence afterwards. Another obvious challenge is that this authority over other objects depends on the application. Each application has its own Head VE as well as its own List of VEs. As a consequence, a Head VE for one application can be in a List of VEs of another application, therefore having a Head VE of its own. This can cause a conflict of authority, when possible circles in the authority chain are created. To avoid this, parental relationships are established only with regards to a certain application.

The motivation behind this relationship again varies a bit from previous relationships. Although the goal of sustaining a knowledge flow among objects remains, the parental relationship introduces the element of authority, which is necessary to some extent, to all communities aiming for autonomy. The existence of a VE with a more central role, able to organize and control the group can help prevent decision deadlocks and solve conflicts when and if they appear. Objects can make decisions for themselves and, when needed, receive decisions made by other objects for them. This results in a community of objects working for themselves, but also for others, towards achieving autonomy, without the need of human interference.

A simple scenario regarding Parental Object Relationship is the following: Smart traffic lights have been installed throughout the city, at various streets. These traffic lights are equipped with sensors able to gather information about traffic on the streets they control. Three traffic lights are part of an sIoT application responsible for Traffic Flow Management of Main Street on Rush Hour. Traffic Light A (TLA) is on Main Street, Traffic Light B (TLB) is on 4th Street and Traffic Light C (TLC) is on 42nd Street. With the use of sensors, TLA is able to acquire information that Main Street has increased traffic. TLA has been set as Head VE for the aforementioned application, which means that it has control over TLB and TLC. TLA is able to coordinate TLB and TLC appropriately by using their IoT services of changing lights from green, to orange, to red. As a result, the three traffic lights are able to decompress traffic on their own, without the need of a traffic officer. A fourth traffic light, Traffic Light D, can be found on the 110th Street. TLD is not part of the same application as TLA-TLC and cannot be thus controlled by TLA.

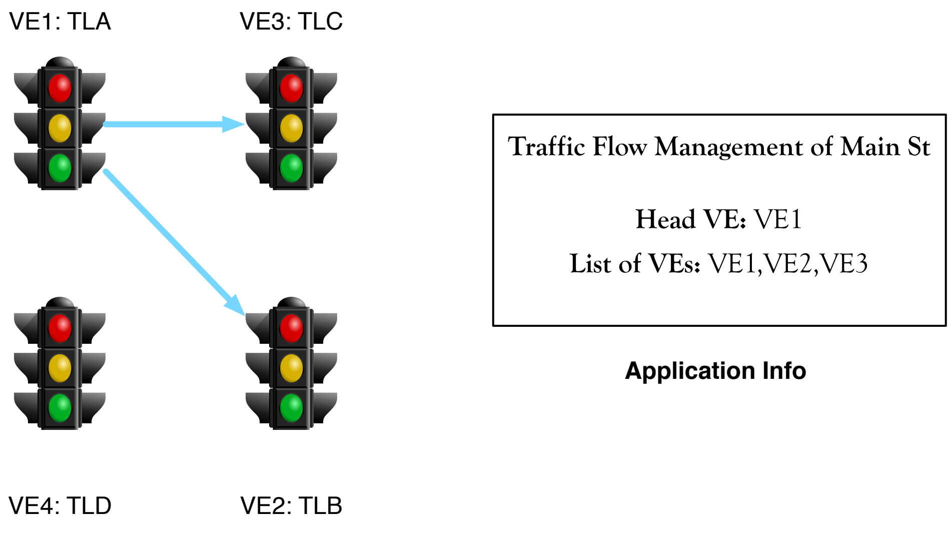


***Figure 4.1.7**: Parental Object Relationship Scenario*

## 4.1.8 Co-Work Object Relationship

The Co-Work Object Relationship connects objects that work on the same application. As it has been previously stated, each sIoT application deploys a number of objects, represented by their Virtual Entities, able to gather information from their environment and other Virtual Entities, gain knowledge from this information in order to make autonomous decisions and provide a number of IoT services. Applications may need a group of objects to form a team, in order to share resources, either in the form of specific knowledge or in the form of IoT services. The co-work relationship is featured prominently in human communities. Workers on a construction site must coordinate and cooperate in building a certain part of a building. Employees in a department store maintain constant communication, querying for inventory and services. Multinational companies have offices on different countries and employees of different nationalities working together on the same company projects. This relationship can also be found in computing, with the emergence of cloud computing. Services that are hosted on the cloud can be hosted on different servers and machines, that work together to support the service.

The importance of applications in this relationship makes the Co-Work Object Relationship an **application based** relationship. This means that applications define the formation of these kinds of relationships between VEs, by providing all the necessary information. The type of an application plays an important part in the nature of this relationship. Applications can be of a fixed size, with a predefined number of VEs involved, or of an evolving size, with VEs joining and leaving the application as needed. Duration of applications can vary as well. Applications can have a predetermined duration or be of open time. Finally, applications may or may not have certain goals and milestones. All the above characteristics make the Co-Work Object Relationship a **dynamic** relationship and of variable **reliability,** depending on the application characteristics. For example, a fixed size and fixed time application will form more reliable Co-Work Object Relationships than an evolving size, open time application. The Co-Work Object Relationship is also a **symmetric** relationship, meaning that if a VE has a Co-Work Object Relationship with another VE, then the same applies for the opposite direction. In terms of Relational Models Theory, this relationship integrates three of the four basic relational structures, **Equality Matching** and **Communal Sharing** and **Authority Ranking** [54].

As mentioned before, this relationship is an application based relationship, which means that applications contain all the necessary information to establish relationships of this kind between VEs. The main property that needs to be examined is the **List of VEs** of the application. This list contains all the VEs that are associated with the application. A Co-Work Object Relationship is formed among all VEs that are included in the List of VEs.

A main challenge of the Co-Work Object Relationship is the dependability on application and the presence of VEs that participate in more than one applications at a time. An easy paradigm to display this problem is to examine a cloud of computers working on multiple applications at once. A certain computer may be hosting two different applications, forming Co-Work Object Relationships with two different groups of other computers. This raises an issue of an exponential increase of Co-Work Object Relationships, jeopardizing navigability in the social internet of things. Therefore, we must consider these types of relationships as fixed with regards to a certain application, to avoid overlap with similar relationships created by other applications.

The motivation for this relationship lies both in the need for knowledge flow among objects with the same purpose, as they operate under the same application, as well as the ability for VEs to share IoT services. The core RMT structure of the Co-Work Object Relationship is Communal Sharing, which calls for cooperation towards a shared goal. This is exactly what makes this relationship so important, objects' ability to identify the common goal and work together towards achieving it, by forming an autonomous community with a purpose.

A simple scenario regarding the Co-Work Object Relationship is the following: A Construction Company has recently purchased various construction machines and has deployed them on the lot of 42nd and 5th Street, where there are plans for a new skyscraper to be built. These construction machines are equipped with smart sensors that allow them to gather information regarding the level of completion of the skyscraper. Construction machines CMA, CMC and CMD are in the list of the application responsible for the completion of the skyscraper. Another construction machine, CMB, is currently offsite and is working on another application. As the List of VEs of the application contains CMA, CMC and CMD (Virtual Entities VE1, VE3 and VE4 respectively), they form Co-Work Object Relationships among them. CMB (Virtual Entity VE2) is not part of the same application and will not form a Co- Work Object Relationship with the other VEs.

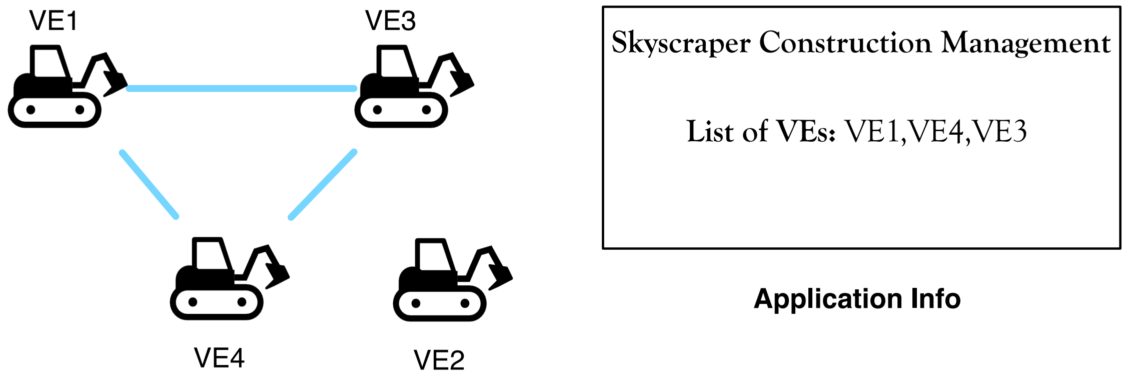


*Figure 4.1.8: Co - Work Object Relationship scenario*

### 4.1.9 Conflict of Interest Object Relationship

The Conflict of Interest Object Relationship is established between objects that may enter a conflict. Conflicts are created when objects vie for the same resources, either in the form of critical information or in the form of vital IoT services that can be offered to a limited number of objects. Conflicts are common in human networks: people interviewing for the same company position; prospective buyers bidding for the same piece of art at an art auction; companies going head to head for market share; countries with different interests and motivations that go to war. Conflicts can also be found in computer networks, when various devices race for the same bandwidth, or even in computer scheduling, where threads and processes aim for the same resources, such as CPU and RAM.

The importance of applications in this relationship makes the Conflict of Interest Object Relationship an **application based** relationship. This means that applications define the formation of these kinds of relationships between VEs, by providing necessary information for their calculation. The type of an application plays an important part in the nature of this relationship. As mentioned before, applications can be of a fixed size, with a predefined number of VEs involved, or of an evolving size, with VEs joining and leaving the application as needed, different time durations and different goals. All the above characteristics make the Conflict of Interest Object Relationship a **dynamic** relationship and of variable **reliability,** as VEs could come and go and conflicts are easily created and resolved, constantly. The Conflict of Interest Object Relationship is also a **symmetric** relationship, meaning that if a VE has a Conflict of Interest Object Relationship with another VE, then the same applies for the opposite direction. In terms of Relational Models Theory, this relationship integrates three of the four basic relational structures, **Equality Matching** and **Market Pricing** and **Authority Ranking.**

As mentioned before, this relationship is an application based relationship, which means that applications contain necessary information to establish relationships of this kind between VEs. The main properties that need to be examined in the application side, is the **List of VEs**, **Importance Table** and the **Financial Package Table** of the application. In fact, the List of VEs is enough to determine conflict of interest among objects. The two tables however are necessary to calculate priority among VEs, in order to ultimately resolve the conflict, with the *Conflict Resolution Method.* VE properties **Domain** and **Financial Package** are also needed to determine VE priority, based on the priorities set by the application itself. Establishing Conflict of Interest Object Relationships is simple: All VEs in the List of VEs form Conflict of Interest Object Relationships among themselves.

The ***Conflict Resolution Method*** is based on the Importance Class and the Financial Package Tables. The Importance Class Table lists domains in order of priority, separated in classes. VEs are then sorted based on their **Domain** property. In case this sorting is not enough, further sorting is made possible based on the Financial Package Table. Similarly to the Importance Class sorting, VEs that tie are further sorted based on their **Financial Package** property. In case of further ties, order between the tied VEs is decided at random.

Challenges arise on the contents and the available options for the two tables, the Importance Class Table and the Financial Package Table. Filling these two tables require user involvement, when setting up a new application. For the first table, all available Domains will be given to the user to sort into classes. This means that Importance is not decided by the social Internet of Things, but by creators of applications. However, it has been observed that importance is subjective and depends on a plethora of outside factors that objects cannot have access to. For the Financial Package Table, the selection of possible packages is left to the creator of the application. A set of default packages can be provided as well, namely the Insufficient, Basic and Premium choices. Another challenge has already been introduced in previous application based relationships and it regards the exponential increase in connections, as VEs can be part of more than one applications. In order to maintain network navigability, this relationship will always be established for one particular application at a time.

So far in the previous relationships, the presence of three of the main Relational Models Theory is prominent; Co - Work Object Relationship highlights Communal Sharing, Parental Object Relationship introduces Authority Ranking and Followers/Followees Object Relationships are based on Equality Matching. The motivation behind the Conflict of Interest Object Relationship, however, is tied to the fourth basic structure of RMT Theory: Market Pricing. Although the ultimate goal of the social Internet of Things is to create autonomous objects that are able to make decisions and cooperate on their own, there is always the possibility that an object must make a decision that puts it in direct conflict with another object. Objects will always attempt to act on decisions they make, based on the knowledge they have acquired, causing deadlocks that can slow down or even break the network. Introducing the Conflict of Interest Object Relationship, along with the Conflict Resolution Method proposed above, will allow objects to move past deadlocks by calculating priorities based on the characteristics of the applications they are committed to.

A simple scenario regarding the Conflict of Interest Object Relationship and the Conflict Resolution Method is the following: A major power failure has created a blackout in an entire building housing offices and other public services. In case of a blackout, an application responsible for running a backup generator has been set up. The backup generator can only provide enough power for 4 of the 6 floors. For each floor, a certain financial package for the application has been purchased. Each floor is equipped with a smart electricity system that is represented in the social internet of things with a Virtual Entity. Each VE has its own Domain and its own Financial Package properties. The application involving all 6 VEs has been configured with an Importance Class Table and a Financial Package Table, containing the default financial package choices of Insufficient for those who have not purchased a sufficient financial package, Basic, for those who have purchased the Basic package and Premium, for those who have purchased a package for higher priority. All 6 VEs form Conflict of Interest Object Relationships among them. Based on the Conflict Resolution Method described above, the VEs are sorted and the first four are supported by the backup generator. Details of the scenario are given in the following figure:

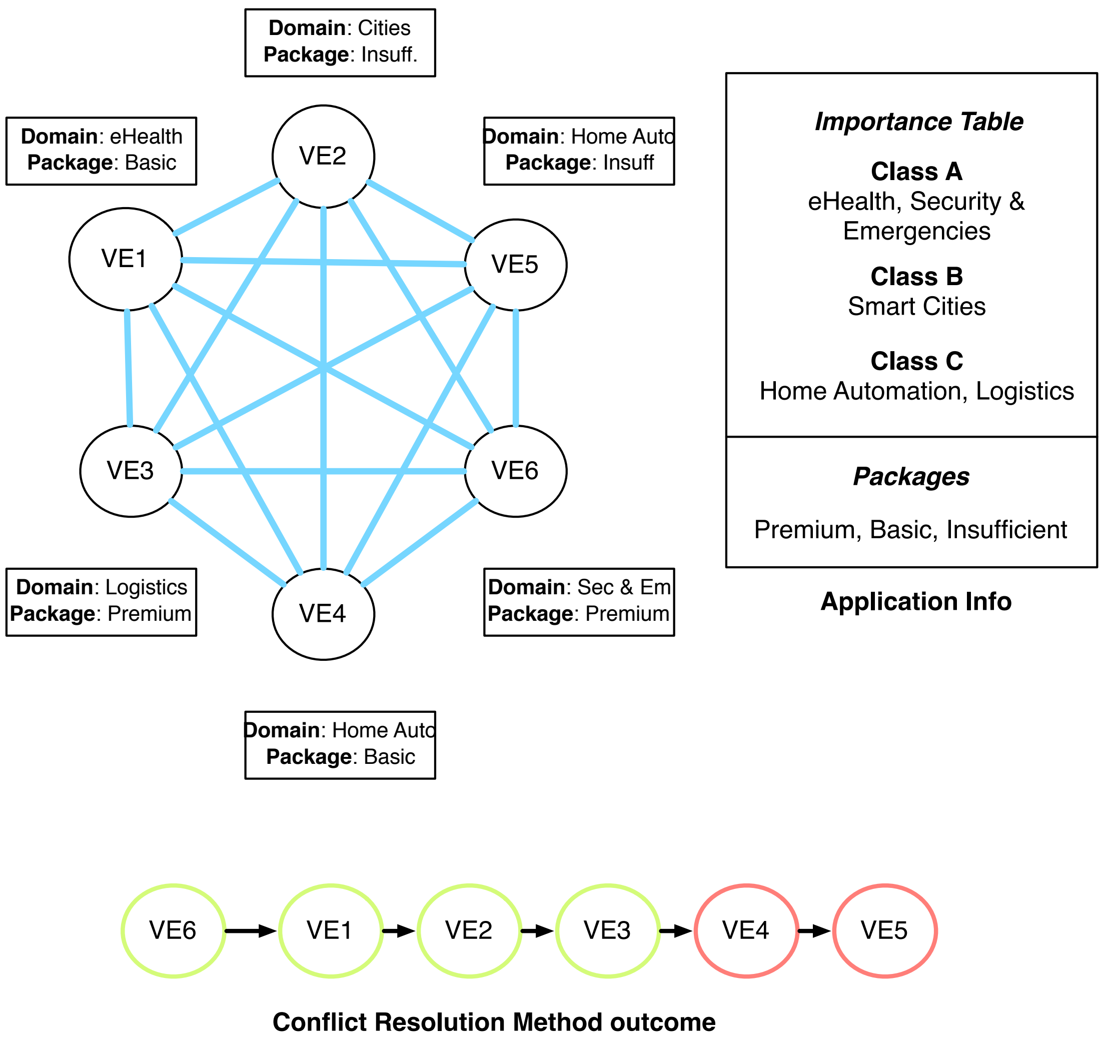


***Figure 4.1.9***: *Conflict of Interest Object Relationship scenario*

## 4.1.10 Replacement Object Relationship

The Replacement Object Relationship is a relationship that can be established between an object and another object that has the capacity to replace it. A replacement object is defined as the object that can perform in an identical way as the object it replaces, making the replacement not having any effects on the social Internet of Things. The idea of replacement can be found in the human network as well. A new employee hired to replace an old employee that has left a company, a soccer player substituting an injured player at a soccer game, a Vice President of a country becoming Acting President when the President resigns. Replacing objects is common practice when noticeable problems in functionality are observed: replacing old batteries in an electronic radio, or changing a flat tire in a car. Replacing objects with humans (and vice versa) is also possible. For example, a traffic officer acts as all traffic lights at an intersection when the lights are malfunctioning. In all cases, a replacement is made in order to maintain a service. The radio is needed for its service of playing music, the car for its service of getting someone from home to work etc. The Replacement Object Relationship aims at maintaing services as well. Objects are needed to replace other objects in order to maintain the offering of IoT services.

The Replacement Object Relationship is also an **application based** relationship, as the formation of these relationships is determined by the application and not the VEs themselves. The type of an application plays an important part in the nature of this relationship. As mentioned before, applications can be of a fixed size, with a predefined number of VEs involved, or of an evolving size, with VEs joining and leaving the application as needed, different time durations and different goals. All the above characteristics make the Replacement Object Relationship a **dynamic** relationship and of varying **reliability,** however, as the nature of replacement relies on VE properties as well, these relationships tend to be more reliable once they are established. The Replacement Object Relationship is also an **asymmetric** relationship, meaning that if a VE has a Replacement Object Relationship with another VE, then the same does not necessarily apply for the opposite direction. In terms of Relational Models Theory, this relationship integrates three of the four basic relational structures, **Equality Matching** and **Communal Sharing** and **Authority Ranking.**

In order for an object to be able to replace another object, it must be able to offer the same IoT services. The main property that is thus needed for the calculation of the Replacement Object Relationship is the **List of IoT Services.** This list contains all the IoT Services that an object can perform. For this relationship to be established, the List of IoT services of the first object must be a subset of the List of IoT Services of the second object. If that is true, then the second object can replace the first object, therefore creating a Replacement Object Relationship. In the event that the two lists are the same, then the two objects can be considered identical. The **List of VEs** provided by the application is needed to identify replacement candidates.

A challenge that all application based relationships share is the consequences of Virtual Entities participating in more than one applications at the same time on network navigability. The amount of possible Replacement Object Relationships for all applications is too high to ensure a scalable network. Therefore, to overcome this, this relationship will always be established for one particular application at a time.

As mentioned before, capacity of a VE to replace another VE is determined by the List of IoT Services. A VE can replace another VE if and only if its List of IoT services contains all the IoT Services found in the relevant list of the second VE. However, ability to efficiently and successfully replace a VE must also be featured by the platform. Replacement VEs have various level of success. When multiple VEs are able to act as replacements, the most efficient and successful one must be chosen first. In order to achieve this, **Replacement Rating** is introduced. Replacement Rating is a rating system that measures replacement efficiency of a VE. Once a replacement is completed, VEs can rate their replacements based on their level of success. This level can be measured in the percentage of completed tasks during the replacement period, or can be manually inputed by the user of the VE. In both cases, a replacement grade is sent to the replacement VE. The Replacement Rating is calculated as the average of all replacement grades that the VE has received. Continuous success in replacing VEs will lead to a higher Replacement Rating. When searching for replacements, available VEs with higher Replacement Ratings will appear first.

The motivation behind the Replacement Object Relationship is essentially the same as with every kind of replacement; to ensure smooth operation. Objects are constantly replaced for maintenance and sustainability reasons, however, so far objects could not be intelligent enough to identify their own replacements. Providing a proper relationship to achieve this will translate to fewer human interference when an object needs to be replaced, as the procedure will be entirely supported by the social internet of things instead.

A simple scenario regarding the Replacement Object Relationship is the following: The city has deployed four new driverless buses that are equipped with sensors to determine their position and when to stop at bus stations, open their doors and take passengers on board. The four buses are represented in the social Internet of Things with their Virtual Entities, VE1, VE2, VE3 an VE4 respectively. Buses VE1 and VE2 are also equipped with a disability door that opens automatically when it detects a disabled person at the bus stop. All four buses are part of an application named Public Transport Automation. When bus VE1 breaks down because of a mechanical failure that cannot be fixed based on the knowledge acquired and is in need of a replacement, it will establish a Replacement Object Relationship with VE2, as this VE has all the IoT Services that it has. Details of the scenario, along with all Replacement Object Relationships that are established among the 4 VEs, are given in the following figure:

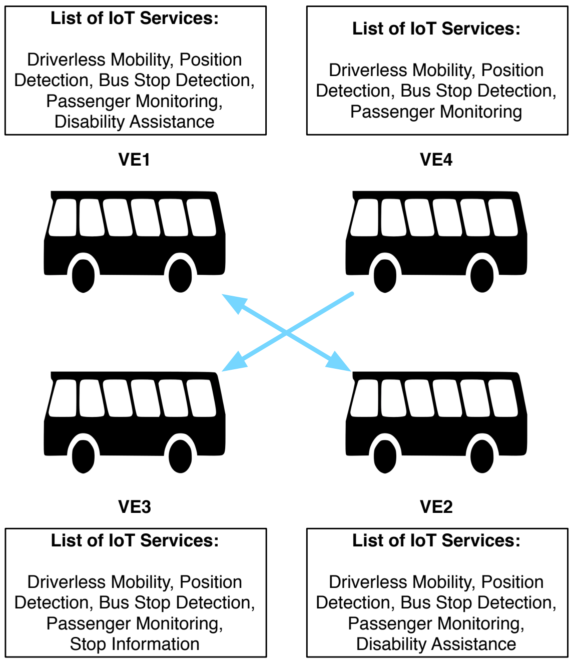

List of IoT Services:
Driverless Mobility, Position Detection, Bus Stop Detection, Passenger Monitoring, Disability Assistance
VE1
List of IoT Services:
Driverless Mobility, Position Detection, Bus Stop Detection, Passenger Monitoring
VE4
VE3
List of IoT Services:
Driverless Mobility, Position Detection, Bus Stop Detection, Passenger Monitoring, Stop Information
VE2
List of IoT Services:
Driverless Mobility, Position Detection, Bus Stop Detection, Passenger Monitoring, Disability Assistance

***Figure 4.1.10***: *Replacement Object scenario*

### 4.1.11 Co-Location Object Relationship

The Co - Location Object Relationship is a relationship that is established between objects that are close to each other. Proximity of objects is defined as the distance between two objects that satisfies a given lower threshold. Location based information relies heavily on the location this type of information has been gathered. This type of information is typically used in applications regarding weather forecasting or geological and geographical surveying, but can also be used in marketing, to offer location based promotions. A notable example of location affecting information is following: If two sensors are placed in two different cities in two different places in the world, for example Athens (Greece) and Boston (MA, USA), then notable differences come up. Because of the different time zones, time reading in Greece will be 7 hours ahead. For example, if the two sensors transmit their time readings at 10:00 UTC, then Greece will transmit 13:00 local time and Boston will transmit  06:00 local time. Sunlight measurements will also be different, as more sunlight will be found in Greece, because it is noon , while the sun will not have fully risen in Boston yet. A final difference can be found in the weather reading, as Boston tends to be colder than Athens.

The Co - Location Object Relationship is a **location based** relationship, as the formation of these relationships is determined by the location of the VEs themselves. Calculation of location is a complex procedure which requires different levels of accuracy, depending on the scope and scale of the application. Applications focused on services among countries require less accuracy than application regarding services in the same room. All the above characteristics make the Co - Location Object Relationship a **dynamic** relationship and of **low reliability**,as it is subject to constant change as location coordinates change, reflecting measurements and statistical errors. The Co - Location Object Relationship is also a **symmetric** relationship, meaning that if a VE has a Co - Location Object Relationship with another VE, then the same applies for the opposite direction. In terms of Relational Models Theory, this relationship integrates three of the four basic relational structures, **Equality Matching** and **Communal Sharing** and **Authority Ranking** [54].

As a location based relationship, the main property that needs to be examined is the **Geolocation** of the VE, especially the **hasGeoLat** and **hasGeoLon** data-type properties. However, as mentioned before, the necessary threshold to determine proximity needs to be set by the application. For that purpose, the **GeoBoundaries** property of the application is needed as well. The GeoBoundaries is a set of coordinates that create a geographical boundary, determining the area and therefore the threshold on distance for the objects to be considered close. All VEs whose coordinates land inside this specific area that is defined by the GeoBoundaries establish Co - Location Object Relationships among them. The **List of VEs** provided by the application is needed to identify proximity candidates.

Accuracy plays an important role in location based relationships. However, it is really difficult to achieve true accuracy, no matter how advanced location calculation algorithms and systems can get. There will always be the margin of error. This is the biggest challenge that this relationship is faced with. Depending on the vastness of the area that the GeoBoundaries cover, issues with accuracy may or may not interfere with the correct calculation of the relationship. An application that covers an entire country inside its GeoBoundaries will not face many problems because of accuracy errors, as the area covered is big enough to outweigh them. On the contrary, an application that needs to confine objects inside a room relies heavily on accurate location data for the Co - Location Object Relationships to be correctly established. Altitude is also an issue. Revisiting the previous example, there needs to be a way to distinguish between objects on different floors. Another location based issue is that VEs are not always stationary. They can be mobile, such as automobiles, buses and vehicles, or portable, such as portable computers. This characteristic creates a constant movement of VEs in space, which requires multiple calculations to determine if its Co - Location Object Relationships can or cannot remain active. A final challenge is one shared with the application based relationships and regards the participation of VEs in multiple applications at once. As mentioned before, this can exponentially increase the number of Co - Location Object Relationships if we consider all application simultaneously, indirectly creating invalid relationships of the same kind. In order to avoid this confusion, this relationship will always be established for one particular application at a time.

The motivation behind the Co - Location Object Relationship lies in the plethora of location based information that can be gathered by an object and may be deemed useful for other objects as well. In order for objects to be fully aware of their surroundings, they must be aware of their environment and be prepared to distinguish information that is only relevant in a particular location. Decisions an object makes can be affected by the location conditions it is in. For example, a smart heater system will distribute heat in a house in a different way if it is in a cold area, than if it were in a hotter area. Objects acquire intelligence by being able to factor location into their decisions. This type of knowledge is important in the knowledge flow among objects in the social internet of things, as other objects in the same situation need to acquire this type of information. A smart heating system in Alaska cannot rely on information from its counterpart in Madrid, as the temperatures are highly different.

A simple scenario regarding the Co - Location Object Relationship is the following: A cab company has deployed a number of smart cabs throughout various countries of the European Union. These smart cabs are able to gather important information about road status and weather conditions in order to increase safety while on the road. There are currently three smart cabs operating in London, UK and another three have been temporarily transferred from London to Madrid, Spain. The cabs in London have gathered information that shows that it is currently raining and visibility is limited because of fog. The cabs in Madrid report a dry and sunny day instead. A new cab is about to be added in the city of London, that will join the application regarding cab services in UK. Based on their geolocation coordinates and the GeoBoundaries of the application, the new cab is able to establish Co - Location Object Relationships with the three cabs currently in London in order to adjust driving to account for the bad weather. Details of the scenario can be found on the following figure.

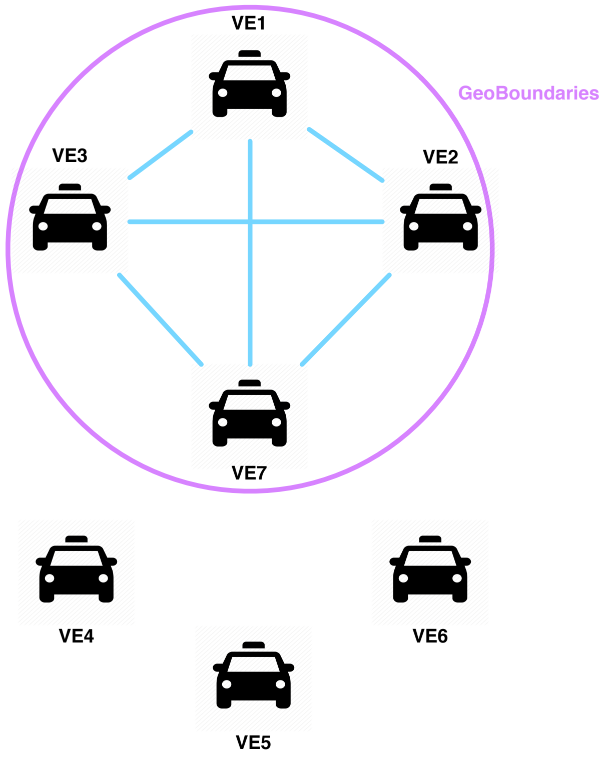


***Figure 4.1.11***: *Co - Location Object Relationship scenario*

The following table summarizes the aforementioned relationships and their characteristics:

| Object Relationships | Ontology Focus | Type | Reliability | Direction | RMT | User Properties | VE properties | Application Properties |
|---|---|---|---|---|---|---|---|---|
| **Ownership** | User based | Static | High | Symmetric | EM, AR | Owner ID | - | - |
| **Usage/ Interaction** | User based | Dynamic | High | Symmetric | EM, AR | - | Simple Users List | - |
| **Domain** | User based | Static | High | Symmetric | CM, EM | - | Domain | - |
| **Follower/ Followee** | VE based | Dynamic | Medium | Asymmetric | CM, EM | - | Followees List | - |
| **Enemy** | VE based | Dynamic | Medium | Asymmetric | EM, MP | - | Enemies List | - |
| **Parental** | VE based | Static | High | Asymmetric | EM, MP, AR | - | - | Head VE List of VEs |
| **Co - Work** | App based | Dynamic | Varying | Symmetric | EM, CM, AR | - | - | List of VEs |
| **Conflict of Interest** | App based | Dynamic | Varying | Symmetric | EM, MP, AR | - | Domain Financial Package | List of VEs Importance Table Financial Package Table |
| **Replacement** | App based | Dynamic | Varying | Asymmetric | EM, CM, AR | - | List of IoT Services Replacement Rating | List of VEs |
| **Co - Location** | Location based | Dynamic | Low | Symmetric | EM, CM, AR | - | Geolocation | GeoBoundaries List of VEs |

## 4.2 Hierarchy of Relational Models

### 4.2.1 Introduction

A major issue of the social Internet of Things is network navigability. It is understood that each user can introduce a number of objects to the network. As the number of objects involved with the network exponentially rises, efficient ways to navigate through a dynamic, complex and continuously growing network are needed. Network navigability and scalability is important in quick and efficient discovery mechanisms, as well as knowledge flow mechanisms. Objects must be able to search through the network in order to discover new information from other objects as well as determine if new relationships must be established, to facilitate this knowledge flow.

The aforementioned object relationships are established between two objects at a time. This means that an object needs to go through all the available VEs in order to check if the relationship requirements are met and then form the relationship. If $n$ is the number of VEs in the network, then this results in a complexity of $O(n^2)$ for all VEs in one relationship, which is a not desirable complexity. In order to overcome this complexity, an hierarchy among the object relationships must be established. Prioritizing among relationships can facilitate machine discovery by eliminating unnecessary VEs from the search range and focusing on specific characteristics. For this purpose, the object relationships are separated into the following four levels:

User Centered Level

The User Centered Level is focused on object relationships where users play a main role in their formation. These relationships are separated into two subcategories, for relationships that are based on user involvement, such as ownership and relationships that are based on user categorization of the VE, such as domain selection. Relationships in this level are often based on fixed properties i.e a certain User ID or domain.The User Centered Level contains the following relationships:

- **Ownership Object Relationship**: This relationship is established between objects owned by the same user.

- **Usage/Interaction Object Relationship**: This Relationship is established between objects used by the same user.

- **Domain Object Relationship**: This Relationship is established between objects that share the same domain.

VE Centered Level

The VE Centered Level is focused on object relationships where Virtual Entities play a main role in their formation. These relationships are based on information that each VE has regarding other VEs, by the use of VE properties that provide clues for direct VE to VE communication. The VE Centered Level contains the following relationships:

- **Followers/Followees Object Relationship**: This relationship is established between objects that share a type of trusted friendship in the form of following .

- **Enemies Object Relationship**: This Relationship is established between objects that are considered hostile.

- **Parental Object Relationship**: This Relationship is established between objects that are ordered under a specific authority.

<u>Application Centered Level</u>

The Application Centered Level is focused on object relationships where Applications play a main role in their formation. These relationships are based on information that each Application has regarding its associated VEs. Relationships in this level are based on a particular application. The Application Centered Level contains the following relationships:

- **Co - Work Object Relationship**: This relationship is established between objects that share the same application .

- **Conflict of Interest Object Relationship**: This Relationship is established between objects that have a conflict of interest when aiming for a limited amount of resources and services.

- **Replacement Object Relationship**: This Relationship is established between objects that can be used as replacement objects, by performing in the same way as the object they replace.

<u>Location Centered Level</u>

The Location Centered Level is focused on object relationships where location plays a main role in their formation. These relationships are solely based on their geographical position, which is defined by their coordinates. The Location Centered Level contains the following relationship:

- **Co - Location Object Relationship**: This relationship is established between objects that are contained inside geographical boundaries defined by their application.

Each level contains specific object relationships that are arranged in a secondary hierarchy inside the level. Hierarchy among the four levels, as well as within each level is decided by the type of information that is available about the objects. Two cases are distinguished: *Fixed Data* and *Generic Data*

| | **Fixed Data** | **Generic Data** |
|---|---|---|
| **Definition** | Absolute | Generic |
| **Hierarchy** | Solid | Fluid |
| **Type** | Static | Dynamic |
| **Reliability** | High | Low |

***Table 4.2.1***: *Differences between the two scenarios*

### 4.2.2 Fixed Data

The most common scenario for the COSMOS platform concerns Fixed Data for the Virtual Entities. Fixed data is the case when all information about the VE is properly inputed, based on the specifications of the platform. Objects in the Fixed Data scenario have valid data on all of the VE properties, giving them an **absolute definition**.

In this scenario, a **solid** hierarchy can be defined. The solid hierarchy model contains specific order among levels, which does not change. Hierarchy within each level is also defined in the same way. Hierarchy of the Fixed Data scenario is as follows:

**Top Level**: User Centered Level

The User Centered Level is the most important level in this scenario. Priority is given over the relationships found in this level. The two subcategories inside the User Centered Level are considered equal, however the first subcategory holds a priority of Ownership over Usage/Interaction Object Relationships.

**Second Level**: VE Centered Level

The User Centered Level is immediately after the User Centered Level. Relationships in this level are usually executed after the execution of one or more relationships from the top level, in order to eliminate VEs from the search. The internal hierarchy of the VE Centered Level contains the Followers/Followees Object Relationship and the Enemies Object Relationship equally at the inner top level and the Parental Object Relationship at the inner bottom level.

**Third Level**: Application Centered Level

The Application Centered Level is immediately after the VE Centered Level. Relationships in this level are usually executed after the execution of one or more relationships from the second level, in order to further eliminate VEs from the search. The internal hierarchy of the Application Centered Level contains the Co - Work Object Relationship and the Conflict of Interest Object Relationship equally at the inner top level and the Replacement Object Relationship at the inner bottom level.

**Bottom Level**: Location Centered Level

The weakest level in the Fixed Data scenario is the Location Centered Level. As the accuracy of the reported locations deeply affect the reliability of these relationships, as well as the relative rarity of searching with geographical restrictions, relationships in this level are executed only when all other levels have failed to produce desired results.

A detailed order of relationships in the Fixed Data scenario can be found in the following figure:

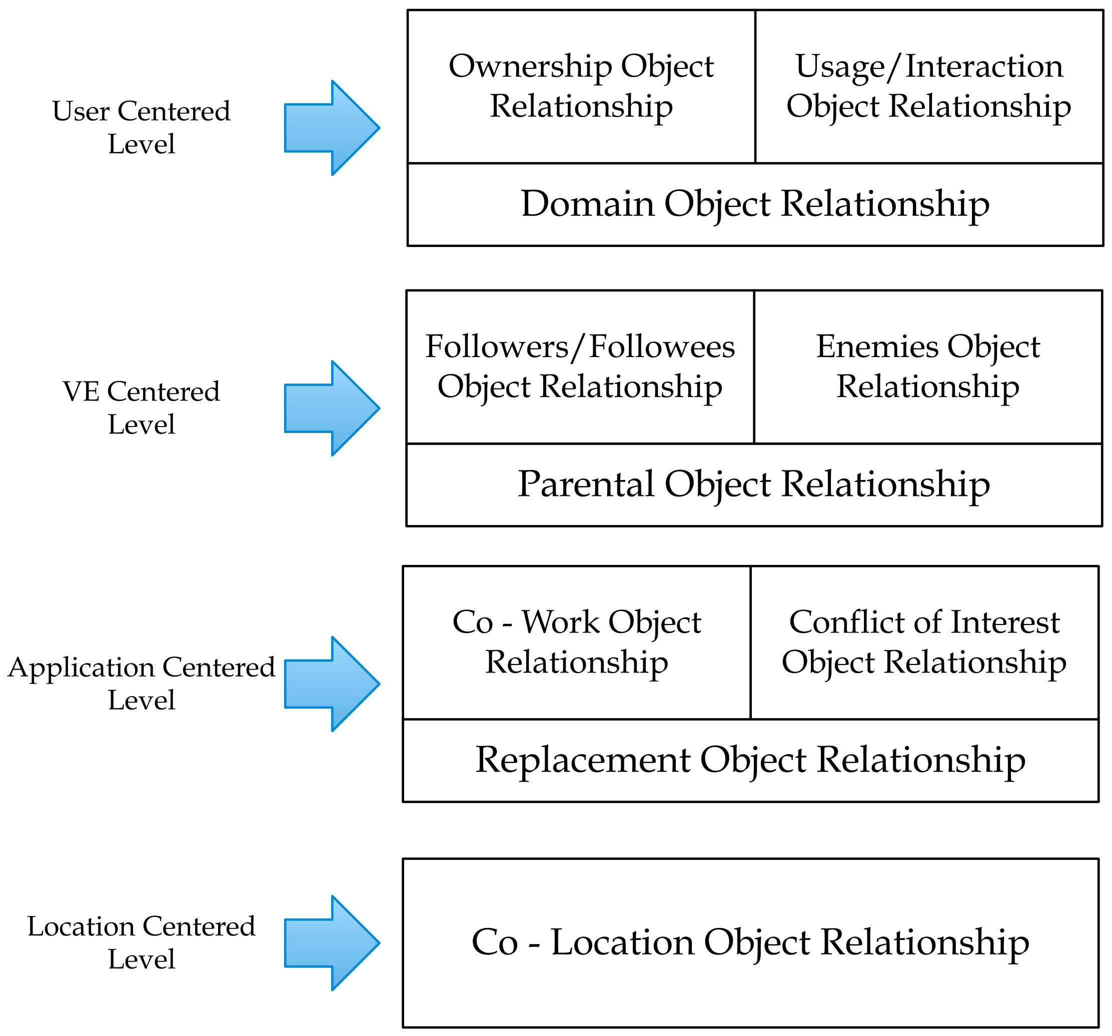
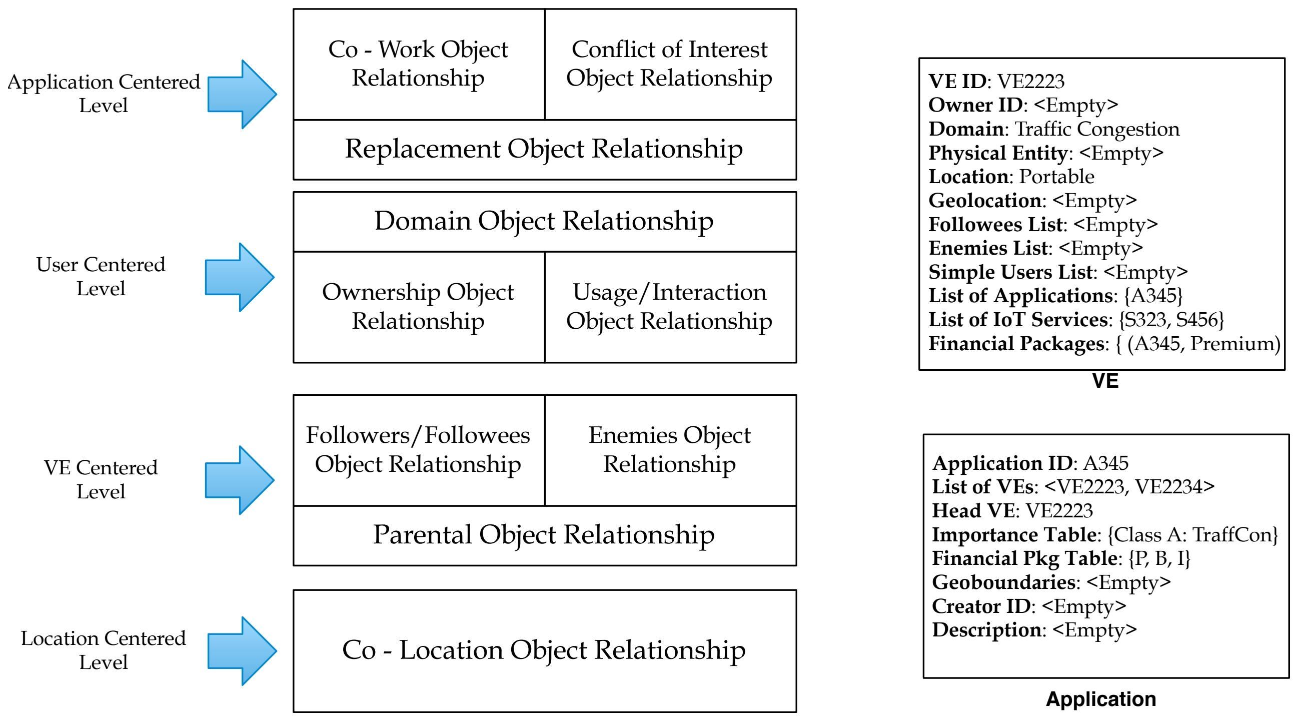

User Centered Level
Ownership Object Relationship
Usage/Interaction Object Relationship
Domain Object Relationship
VE Centered Level
Followers/Followees Object Relationship
Enemies Object Relationship
Parental Object Relationship
Application Centered Level
Co - Work Object Relationship
Conflict of Interest Object Relationship
Replacement Object Relationship
Location Centered Level
Co - Location Object Relationship

***Figure* 4.2.2**: *Fixed Data scenario Hierarchy*

### 4.2.3 Generic Data

As users are responsible with appropriately describing their objects and carefully filling out the VE properties, user factor plays an important role in shaping the VE. Nonetheless, some information can be incomplete or not properly completed during this process for various reasons. This can result in a VEs that have a **generic definition**.

As information on VEs is more generic and not easily interpreted, a **fluid** hierarchy must be defined. The fluid hierarchy model does not specify an absolute order among levels. On the contrary, hierarchy among levels is defined based on the most reliable and well defined information that can be provided by the VEs. For example, if no information is provided for the owner, but information in the List of Followees is valid, then the VE Centered Level acquires a higher position over the User Centered Level. Hierarchy between relationships within a level is determined in a similar fashion. For example, If no information is available for the owner of the VE, but the List of Simple Users is given, then the Usage/Interaction Object Relationship acquires higher priority over the Ownership Object Relationship. In the Generic Data scenario, fluidity of hierarchy also means that this hierarchy is dynamic and subject to changes as information about the VE is inserted or deleted. An example of a fluid hierarchy based on VE information can be found in the following figure:


Application Centered Level
Co - Work Object Relationship
Conflict of Interest Object Relationship
Replacement Object Relationship
User Centered Level
Domain Object Relationship
Ownership Object Relationship
Usage/Interaction Object Relationship
VE Centered Level
Followers/Followees Object Relationship
Enemies Object Relationship
Parental Object Relationship
Location Centered Level
Co - Location Object Relationship
VE
Application

**VE ID**: VE2223
**Owner ID**: <Empty>
**Domain**: Traffic Congestion
**Physical Entity**: <Empty>
**Location**: Portable
**Geolocation**: <Empty>
**Followees List**: <Empty>
**Enemies List**: <Empty>
**Simple Users List**: <Empty>
**List of Applications**: {A345}
**List of IoT Services**: {S323, S456}
**Financial Packages**: { (A345, Premium)

**Application ID**: A345
**List of VEs**: <VE2223, VE2234>
**Head VE**: VE2223
**Importance Table**: {Class A: TraffCon}
**Financial Pkg Table**: {P, B, I}
**Geoboundaries**: <Empty>
**Creator ID**: <Empty>
**Description**: <Empty>

***Figure 4.2.3***: *Generic Data scenario Hierarchy*

# Chapter 5: Network Simulation and Results

## 5.1 Implementation

### 5.1.1 Introduction

The tool that was used for data analysis is **Gephi**[56]. Gephi is an interactive visualization and exploration platform for networks and complex systems, which can also support dynamic and hierarchical graphs. Gephi is open-source and free and can run on Windows, Linux and OS X. Gephi is based on NetBeans UI, has a built-in 3D rendering engine and is customizable by plugins. Gephi was selected for its plethora of applications and metrics such as centrality metrics, density, path length and more. The platform was chosen as it can support multiple formats such as GDF, GraphML (NodeXL), GML, NET (Pajek), GEFX and more.

Data is organized as follows: A GraphML file was created for each object relationship. Each node represents a Virtual Entity and each edge between nodes represents the object relationship between the selected Virtual Entities. An extra GraphML file was created containing all object relationships, representing the complete form of the social network among Virtual Entities. These files were then imported to Gephi for network analysis such as exporting metrics and visualization of the network.

Some of the **layout algorithms** offered in Gephi include:

- **Force Atlas 2**: A force-directed layout close to other algorithms used for network spatialization. Force Atlas 2 Is a continuous algorithm, that allows graph manipulation during rendering. It has a linear-linear model, features a unique adaptive convergence speed and proposes summarized settings. It also features a Barnes Hut optimization. [57]

Force Atlas 2 features these settings:

1) **Scaling**: How much repulsion you want. More makes a more sparse graph.
2) **Gravity**: Attracts nodes to the center. Prevents islands from drifting away.
3) **Dissuade Hubs**: Distributes attraction along outbound edges. Hubs attract less and thus are pushed to the borders.
4) **LinLog mode**: Switch ForceAtlas' model from lin-lin to lin-log. Makes clusters more tight.
5) **Prevent Overlap**: Use only when spatialized. Should not be used with "Approximate Repulsion"
6) **Tolerance** (**speed**): How much swinging you allow. Above 1 discouraged. Lower gives less speed and more precision.
7) **Approximate Repulsion**: Barnes Hut optimization: $n^2$ complexity to n.ln(n) ; allows larger graphs.
8) **Approximation**: Theta of the Barnes Hut optimization.
9) **Edge Weight Influence**: How much influence you give to the edges weight. 0 is "no influence" and 1 is "normal".

- **Fruchterman Reingold**: a force-directed layout algorithm. The idea of a force directed layout algorithm is to consider a force between any two nodes. In this algorithm, the nodes are represented by steel rings and the edges are springs between them. The attractive force is analogous to the spring force and the repulsive force is analogous to the electrical force. The basic idea is to minimize the energy of the system by moving the nodes and changing the forces between them. For more details refer to the Force Directed algorithm. In this algorithm, the sum of the force vectors determines which direction a node should move. The step width, which is a constant determines how far a node moves in a single step. When the energy of the system is minimized, the nodes stop moving and the system reaches it's equilibrium state. The drawback of this is that if we define a constant step width, there is no guarantee that the system will reach equilibrium at all. T.M.J. Fruchterman and E.M. Reingold introduced a "global temperature" that controls the step width of node movements and the algorithm's termination. The step width is proportional to the temperature, so if the temperature is hot, the nodes move faster (i.e, a larger distance in each single step). This temperature is the same for all nodes, and cools down at each iteration. Once the nodes stop moving, the system terminates. [58]

Some of the metrics used for network analysis are the following:

- **Average Degree**
- **Average Weighed Degree**
- **Distance**: The average graph-distance between all pairs of nodes. Connected nodes have graph distance 1. The diameter is the longest graph distance between any two nodes in the network.
- **Betweennness Centrality**: Measures how often a node appears on shortest paths between nodes in the network
- **Closeness Centrality**: The average distance from a given starting node to all other nodes in the network.
- **Eccentricity**: The distance from a given starting node to the farthest node from it in the network.
- **Density**: Measures how close the network is to complete. A complete graph has all possible edges and density equal to 1.
- **HITS**: Computes two separate values for each node. The first value, **Authority**, measures how valuable information stored at that node is. The second, **Hub**, measures the quality of the nodes links.
- **Modularity**: Community detection algorithm
- **PageRank**: Ranks nodes "pages" according to how often a user following links will non-randomly reach the node "page.
- **Connected Components**: Determines the number of connected components in the network.
- **Clustering Coefficient**: The clustering coefficient, along with the mean shortest path, can indicate a "small-world" effect. It indicates how nodes are embedded in their neighborhood. The average gives an overall indication of the clustering in the network.
- **Eigenvector Centrality**: A measure of node importance in a network based on a node's connections.

A notable issue with Gephi is that parallel edges are not supported. On the contrary, any parallel edges that exist on the network are converted into a single edge with weight equal to the sum of parallel edges. For this reason, graphs with parallel edges will vary from other graphs, as all edges will be weighed. A graph with no parallel edges will have edge weight equal to 1. This issue is mostly observed in the complete representation of the network. Metrics take into consideration the weight of edges.

### 5.1.2 Cases Description

A series of facts about the nature of the data are presented below:

a) Each Virtual Entity(VE) has exactly one **Owner**.
b) Each VE belongs to exactly one **Domain**.
c) Each VE participates in exactly one **Application**.
d) The **total number of users** is 25% of the Total number of VEs
e) The **total number of possible Domains** is 8.
f) The **total number of applications** is 2 and VEs are distributed evenly between these two Applications.
g) The **maximum number of users** for any VE is set to 60% of the total number of users.
h) The **maximum number of followees/enemies** for any VE is set to 60% of the total number of users.
i) Only the first application presents issues of **Conflict**.

In order to reduce the size of the social network of the Virtual Entities, the *Replacement* and the *Co - Location* Object Relationships are omitted from the network analysis.

## 5.2 Case 1: 100 Virtual Entities

### 5.2.1 Statistics for Case 1

The following table summarizes the basic statistics for Case1:

| Case 1 | |
|---|---|
| **Total Nodes** | 100 |
| **Total Users** | 25 |
| **Domains** | 8 |
| **Applications** | 2 |
| **Max Friends/Enemies** | 25 |
| **Ownership Relationships** | 223 |
| **Usage/Interaction Relationships** | 37129 |
| **Domain Relationships** | 632 |
| **Followers Relationships** | 21656 |
| **Enemies Relationships** | 14618 |
| **Parental Relationships** | 98 |
| **Co - Work Relationships** | 2450 |
| **Conflict Relationships** | 1225 |
| **Total Relationships** | **78031** |

***Table 5.2.1**: Case 1 Statistics*

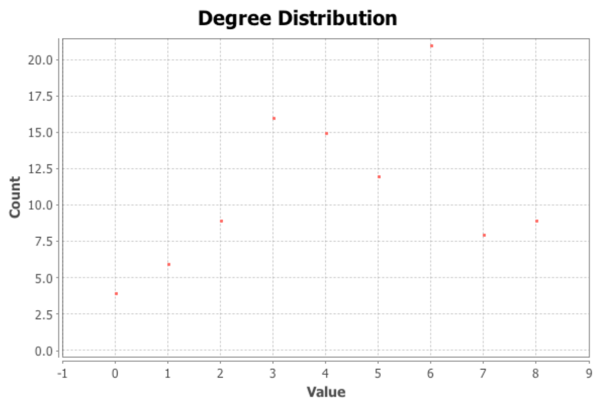
### 5.2.2 Ownership Object Relationship

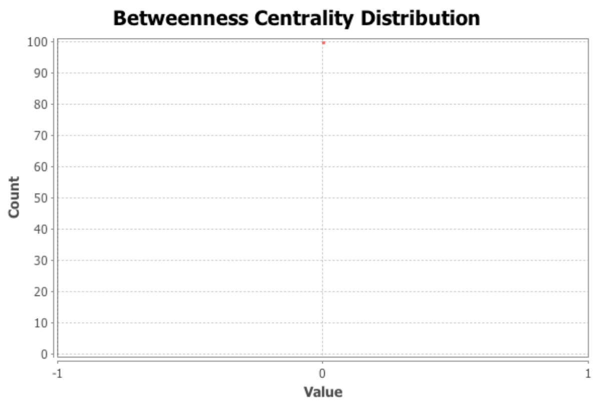
| **Reports** | **Average Degree** | | | |
|---|---|---|---|---|
| **Degree** | 4.460 | | | |
| | Diameter | Radius | Avg. Path Length | #. of Shortest Paths |
| **Graph Distance** | 1 | 0 | 1.0 | 446 |
| | Density | | | |
| **Graph Density** | 0.045 | | | |
| | Modularity | | Modularity with resolution | #. of Communities |
| **Modularity** | 0.913 | | 0.913 | 24 |
| | Number of Weakly Connected Components | | | |
| **Connected Components** | 24 | | | |
| | Avg. Clustering Coefficient | | Total Triangles | |
| **Clustering Coefficient** | 1.000 | | 334 | |

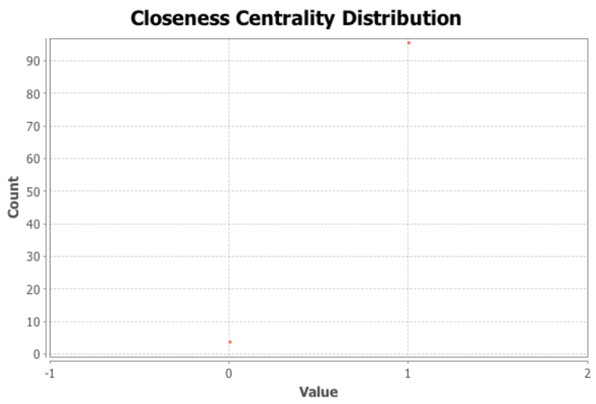
***Table 5.2.2**: Summary of Network Analysis Reports for Ownership Object Relationships of Case 1*

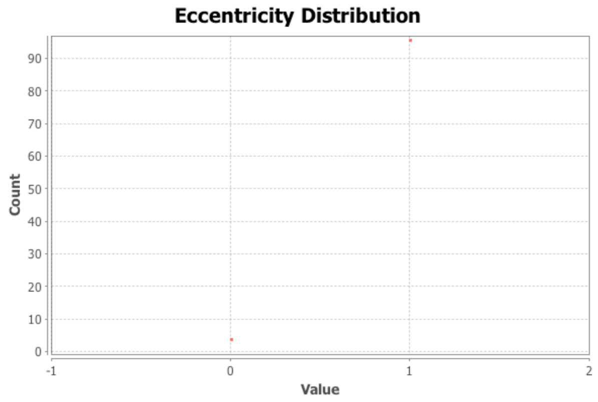
Network analysis of the ownership object relationships immediately reveals useful information about the network. An average degree below 5 shows us that VEs are immediately connected to a small amount of VEs, making the relationship very **specific**. This means that using this relationship is preferred when searching for particular VEs. Such is, however, the nature of this relationship as well. Given the fact that a user usually owns a small amount of objects, the knowledge flow that is created among these VEs is contained in a small group. The low density of the network of ownership relationships is also an indication of a **sparse** network.

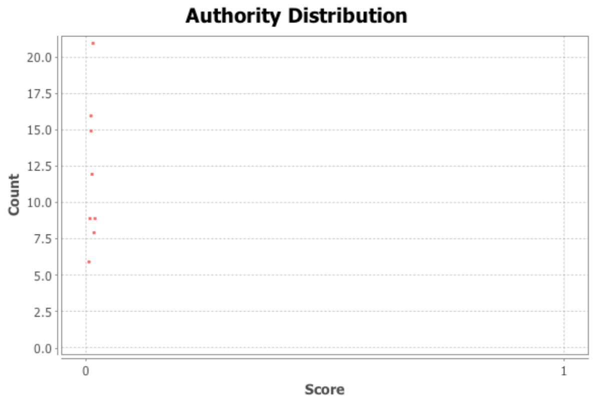
The most interesting piece of information that can be gathered both by the metrics reports and the visualization of the network, regards the number of communities in the network. Both the number of the communities and the number of weakly connected components can lead us to the **number of owners** of the network of Virtual Entities. Each community represents a group of VEs which share the same owner. Based on the statistics of Case 1, we can deduce that each User is also an owner of at least one VE in the network, as the number of Total Users is equal to the number of detected communities, ergo the owners.

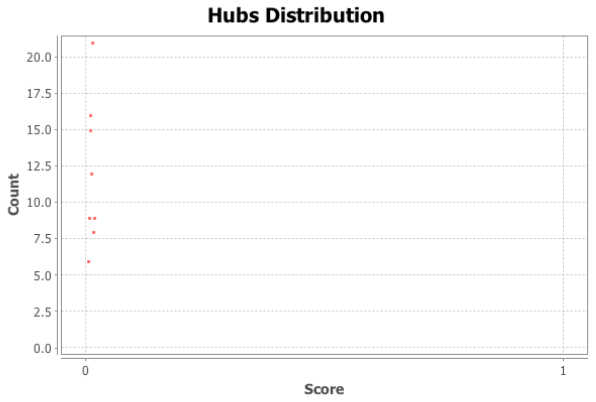

Degree Distribution
Count
Value

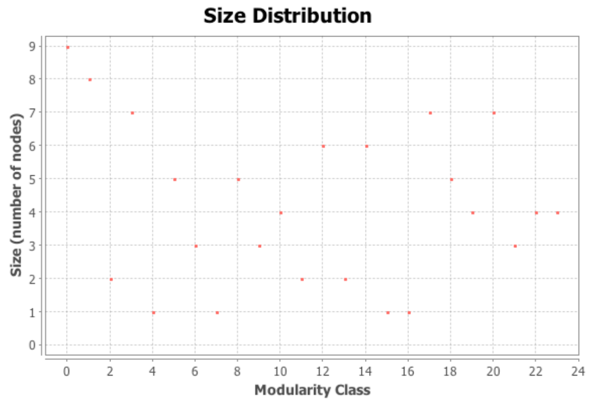

Betweenness Centrality Distribution
Count
Value

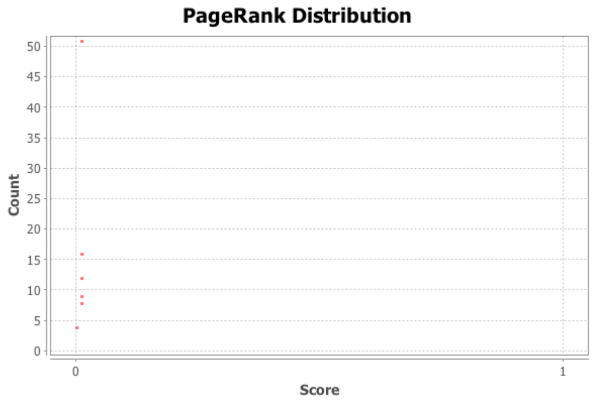

Closeness Centrality Distribution
Count
Value

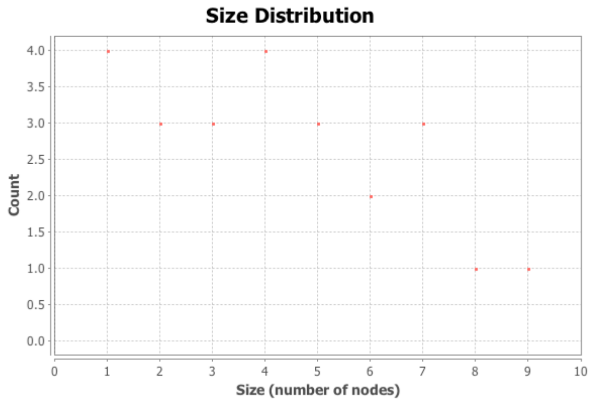

Eccentricity Distribution
Count
Value

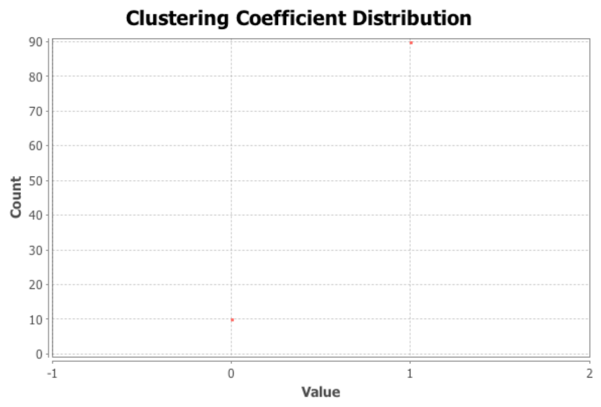

Authority Distribution
Count
Score


Hubs Distribution
Count
Score


Size Distribution
Size (number of nodes)
Modularity Class


PageRank Distribution
Count
Score


Size Distribution
Count
Size (number of nodes)



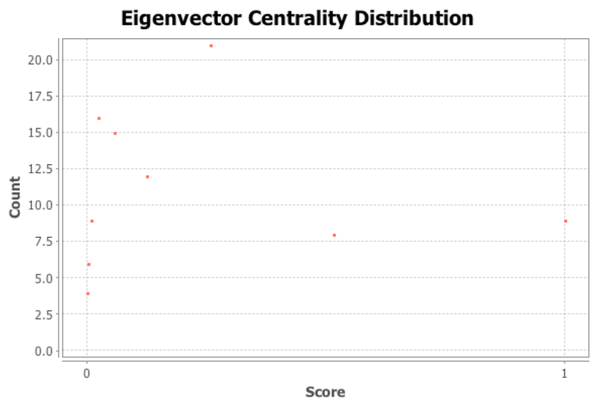


***Figure***: *Network Analysis for Ownership Object Relationships of Case1*
*From top to bottom: Degree Distribution, Betweenness Centrality Distribution, Closeness Centrality Distribution, Eccentricity Distribution, Authority Distribution, Hubs Distribution, Modularity, PageRank Distribution, Connected Components Distribution, Clustering Coefficient Distribution, Eigenvector Centrality Distribution*

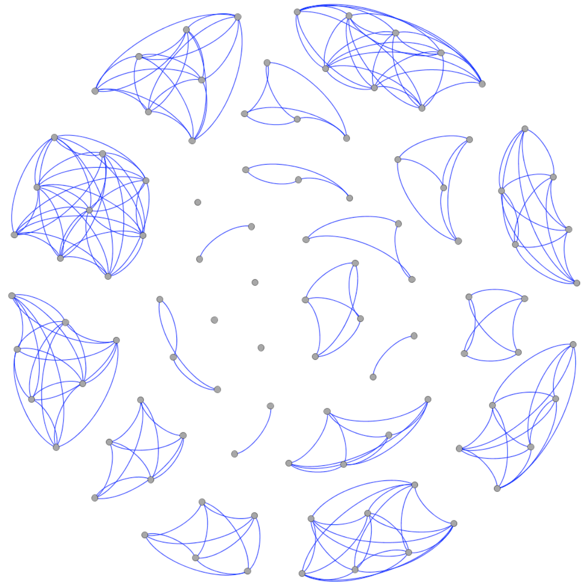


***Figure***: *Ownership Object Relationship Graph*

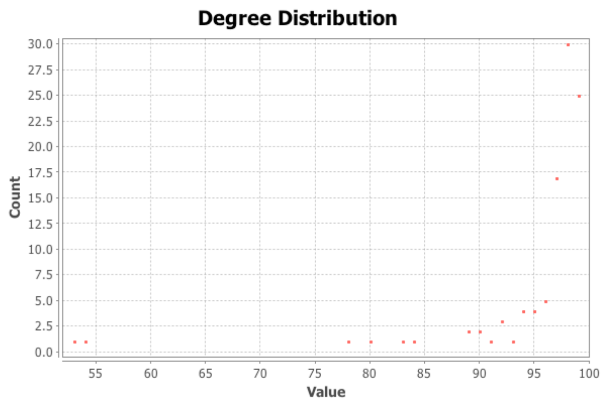
### 5.2.3 Usage/Interaction Object Relationship

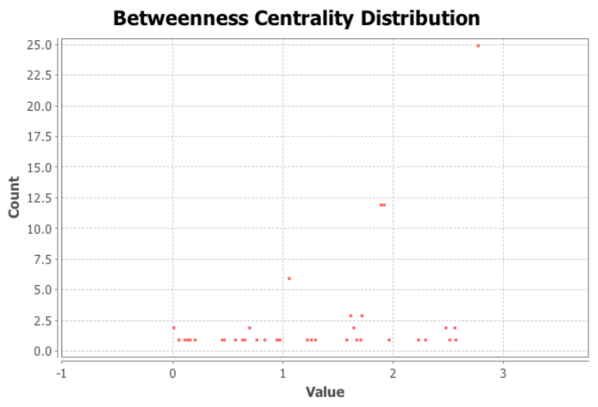
| **Reports** | **Average Degree** | | | |
|---|---|---|---|---|
| **Degree** | 95.500 | | | |
| | Diameter | Radius | Avg. Path Length | #. of Shortest Paths |
| **Graph Distance** | 2 | 1 | 1.03535353535354 | 9900 |
| | Density | | | |
| **Graph Density** | 0.965 | | | |
| | Modularity | Modularity with resolution | #. of Communities | |
| **Modularity** | 0.010 | 0.010 | 3 | |
| | Number of Weakly Connected Components | | | |
| **Connected Components** | 1 | | | |
| | Avg. Clustering Coefficient | Total Triangles | | |
| **Clustering Coefficient** | 0.975 | 147382 | | |

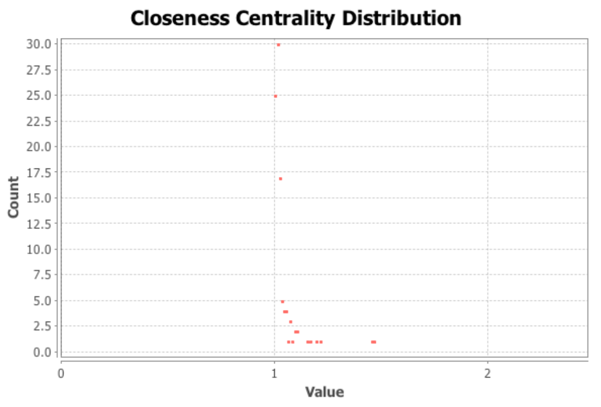
***Table 5.2.3**: Summary of Network Analysis Reports for Usage/Interaction Object Relationships of Case 1*

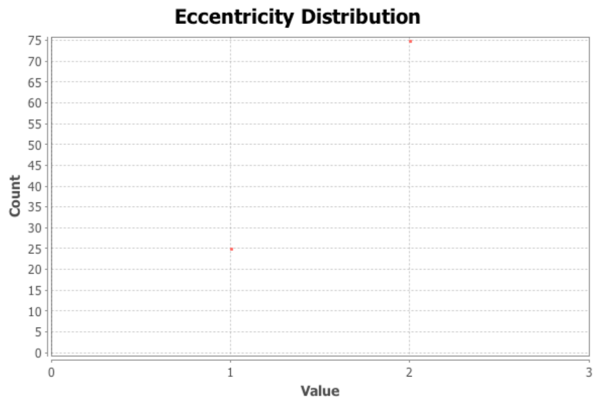
Network analysis of the usage/interaction object relationships immediately reveals useful information about the network. A complete contrast of the ownership relationship, as witnessed by the visualization of the graph, this relationship comprises of much more edges than the ownership relationship. An average degree close to 100 shows us that VEs are immediately connected to a vast amount of VEs, making the relationship **generic**. This is the result of multiple users for each VE. As the number of users is less than the number of VEs, the appearance of similar groups of users on each VEs eventually leads to an almost complete network. The high density of the network of usage/interaction relationships is also an indication of a **dense** network.

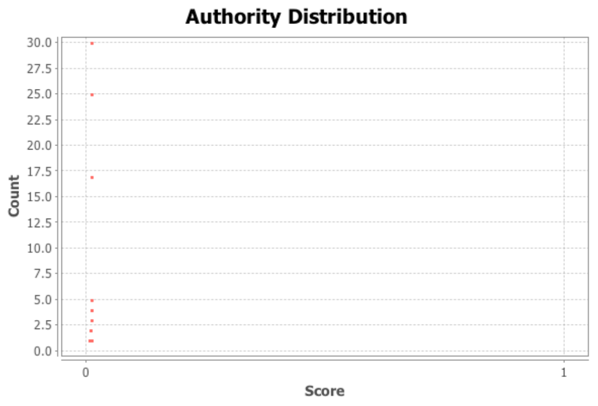
The difference between this relationship and the ownership relationship is also evident in the visualizations of the networks. While communities are easily spotted in the latter, betraying the number of users, the former relationship gives no clues about the number of users. On the contrary, the density of the network is reinforced by the visualization of the network.

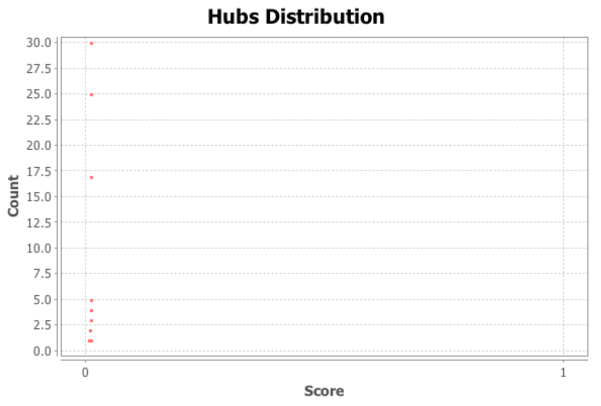

Degree Distribution
Count
Value
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0
55
60
65
70
75
80
85
90
95
100

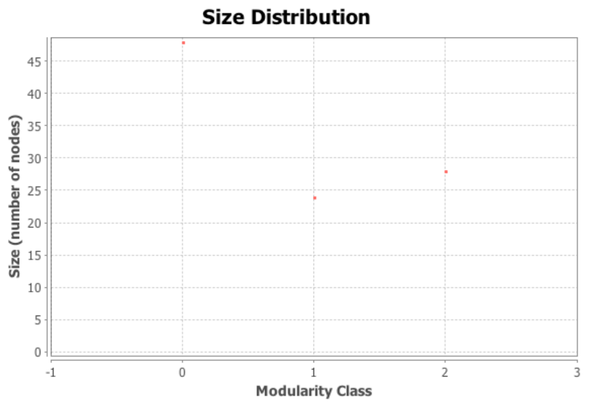

Betweenness Centrality Distribution
Count
Value
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
-1
0
1
2
3

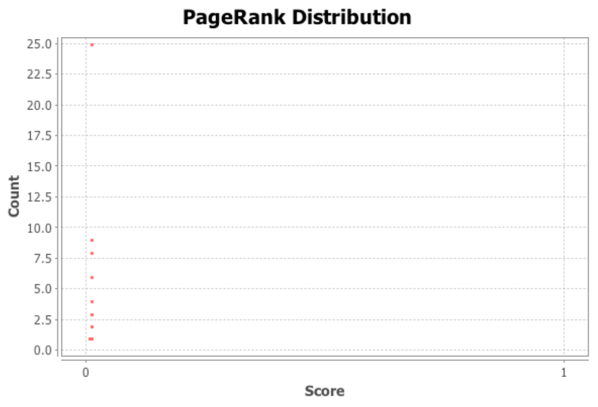

Closeness Centrality Distribution
Count
Value
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0
0
1
2

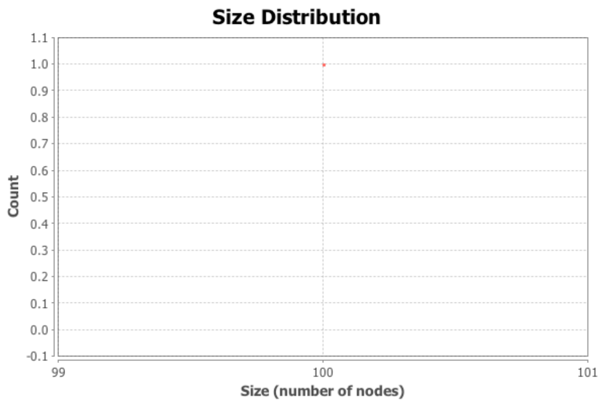

Eccentricity Distribution
Count
Value
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
0
1
2
3


Authority Distribution
Count
Score
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0
0
1


Hubs Distribution
Count
Score
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0
0
1


Size Distribution
Size (number of nodes)
Modularity Class
0
5
10
15
20
25
30
35
40
45
-1
0
1
2
3


PageRank Distribution
Count
Score
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
0
1


Size Distribution
Count
Size (number of nodes)
-0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1.1
99
100
101

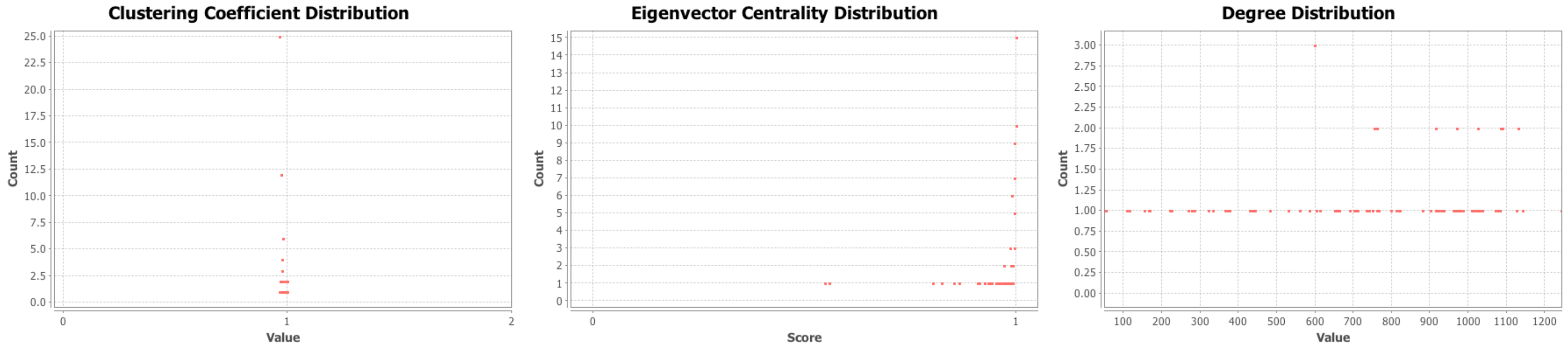


***Figure***: *Network Analysis for Usage Object Relationships of Case1*
*From top to bottom: Degree Distribution, Betweenness Centrality Distribution, Closeness Centrality Distribution, Eccentricity Distribution, Authority Distribution, Hubs Distribution, Modularity, PageRank Distribution, Connected Components Distribution, Clustering Coefficient Distribution, Eigenvector Centrality Distribution*

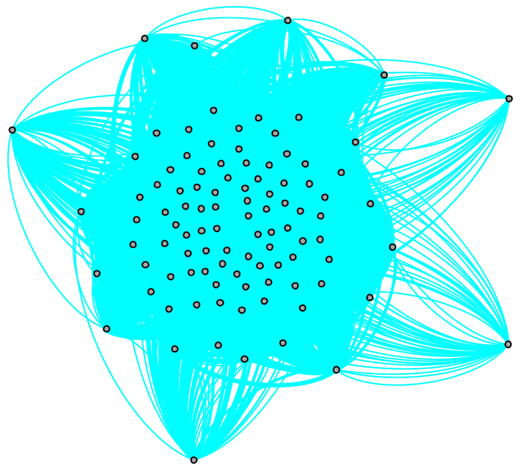


***Figure***: *Usage/Interaction Object Relationship Graph*

### 5.2.4 Domain Object Relationship

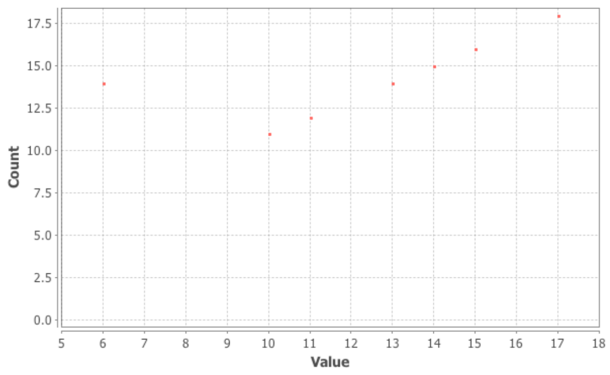
| **Reports** | **Average Degree** | | | |
|---|---|---|---|---|
| **Degree** | 12.640 | | | |
| | Diameter | Radius | Avg. Path Length | #. of Shortest Paths |
| **Graph Distance** | 1 | 1 | 1.0 | 1264 |
| | Density | | | |
| **Graph Density** | 0.128 | | | |
| | Modularity | Modularity with resolution | #. of Communities | |
| **Modularity** | 0.836 | 0.836 | 8 | |
| | Number of Weakly Connected Components | | | |
| **Connected Components** | 8 | | | |
| | Avg. Clustering Coefficient | Total Triangles | | |
| **Clustering Coefficient** | 1.000 | 2650 | | |

***Table 5.2.4****: Summary of Network Analysis Reports for Domain Object Relationships of Case 1*

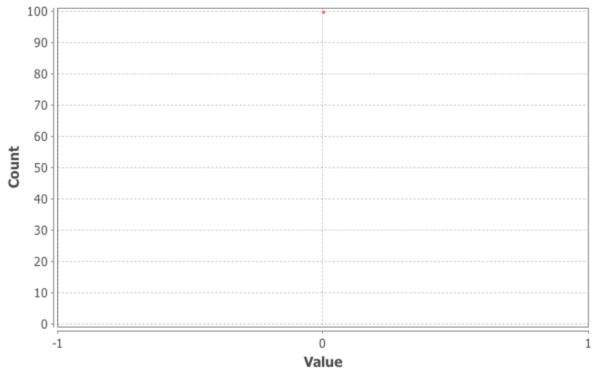
Network analysis of the domain object relationships immediately reveals useful information about the network. An average degree over 12 shows us that VEs are immediately connected to a bigger amount of VEs than through the ownership relationship. However, the number is noticeably low, making the relationship very **specific**. This means that using this relationship is preferred when searching for particular VEs. Given the fact that the number of domains is much lower than the number of VEs in the network, this knowledge flow is comparable to the one maintained with the ownership relationship. The low density of the network of domain relationships is also an indication of a **sparse** network. In fact, the resulting network is even more sparse than the ownership relationship equivalent.

The most interesting piece of information that can be gathered both by the metrics reports and the visualization of the network, regards the number of communities in the network. Both the number of the communities and the number of weakly connected components can lead us to the **number of domains** of the network of Virtual Entities. Each community represents a group of VEs which share the same domain. The sum of these groups is of course the total number of domains in the network. This can also be verified by the statistics of the network of Case1, which gives 8 domains, same as the number of communities.

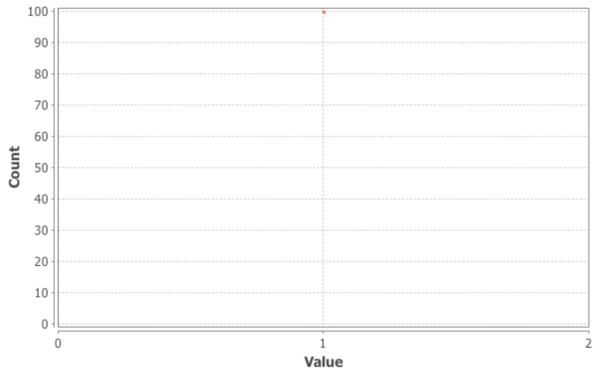
Degree Distribution



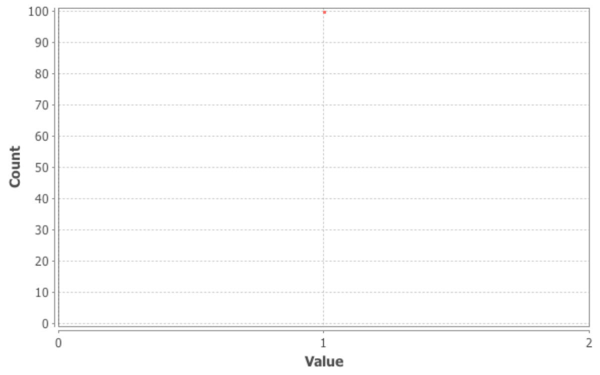
Betweenness Centrality Distribution



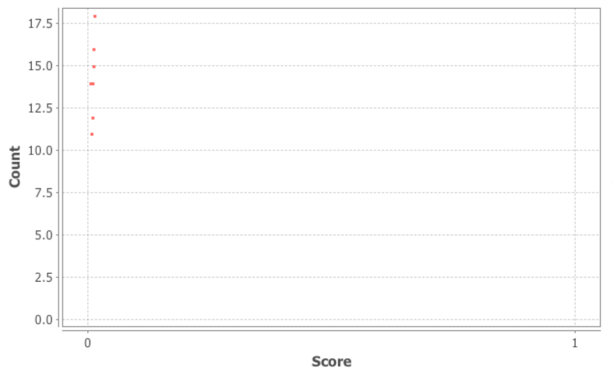
Closeness Centrality Distribution



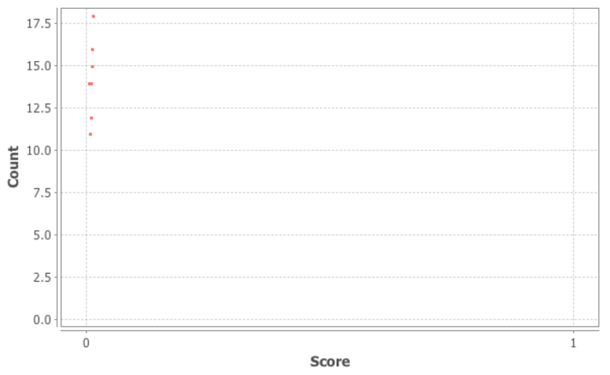
Eccentricity Distribution



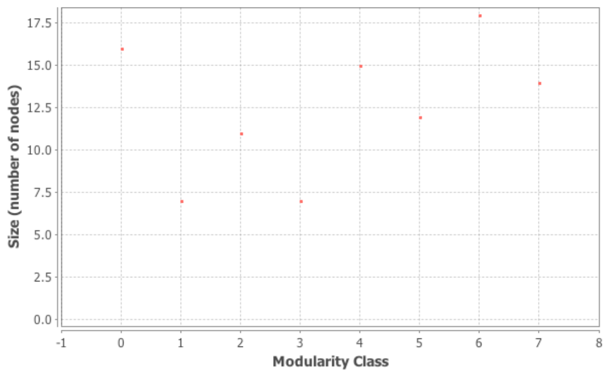
Authority Distribution



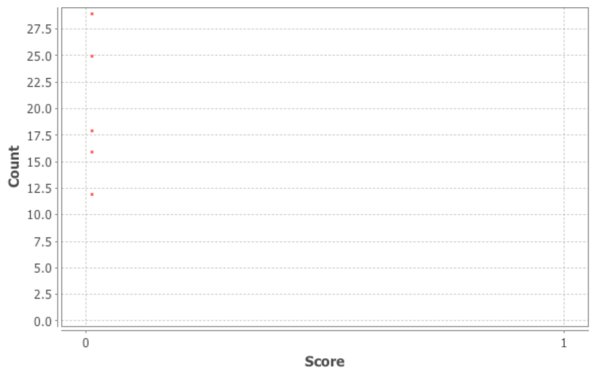
Hubs Distribution



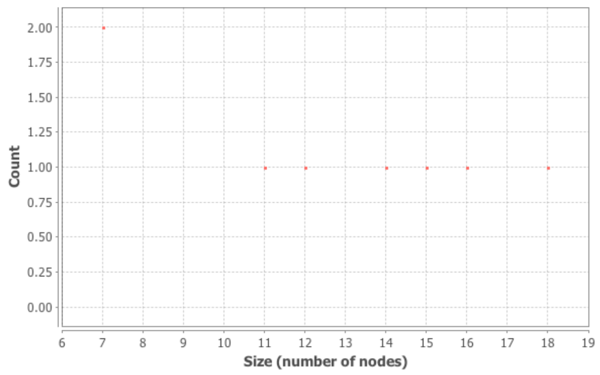
Size Distribution

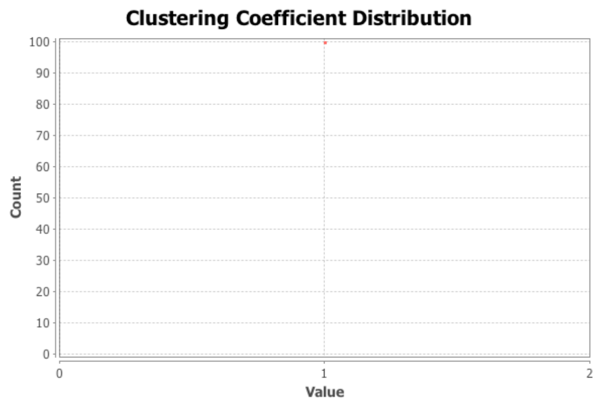


PageRank Distribution



Size Distribution





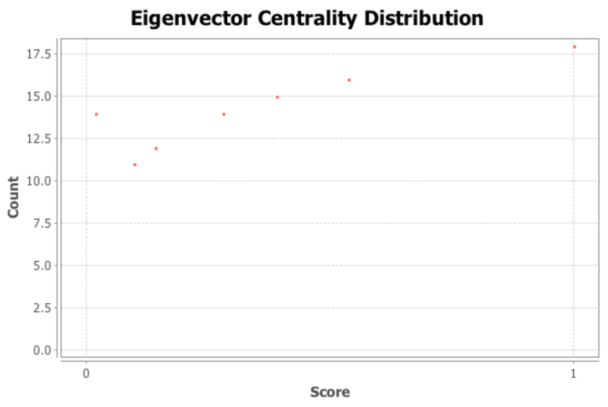


***Figure***: *Network Analysis for Domain Object Relationships of Case1*
*From top to bottom: Degree Distribution, Betweenness Centrality Distribution, Closeness Centrality Distribution, Eccentricity Distribution, Authority Distribution, Hubs Distribution, Modularity, PageRank Distribution, Connected Components Distribution, Clustering Coefficient Distribution, Eigenvector Centrality Distribution*

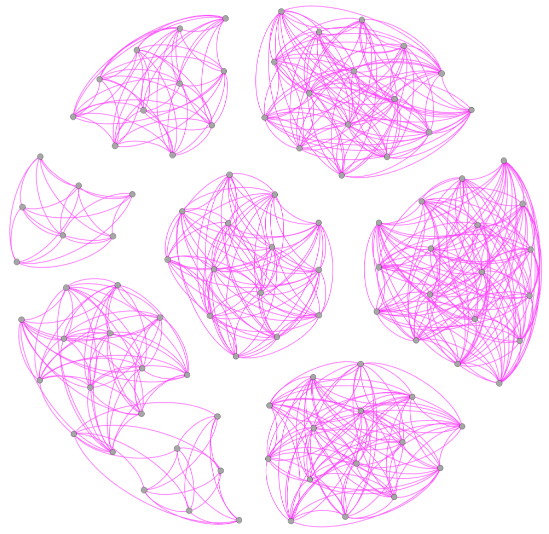


***Figure***: *Domain Object Relationship Graph*

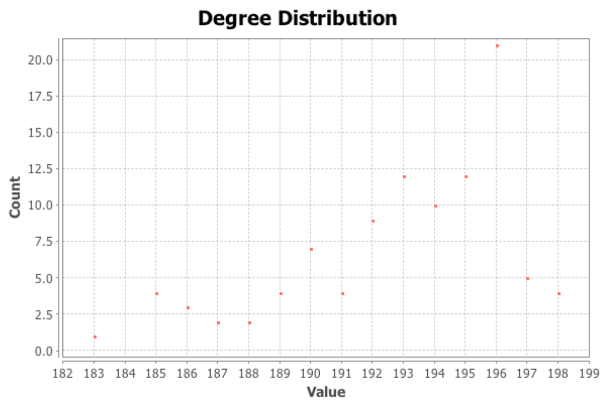
## 5.2.5 Followers and Enemies Object Relationships

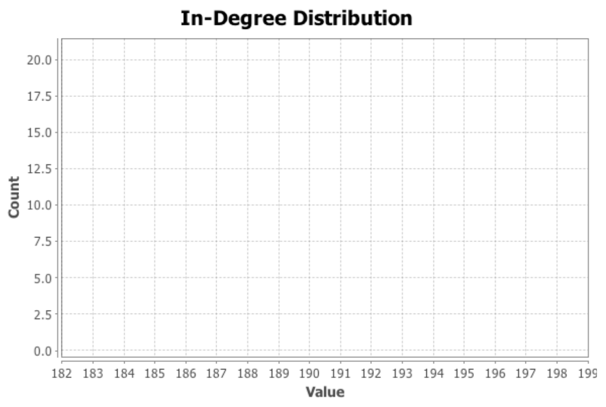
| Reports | Average Degree | | | |
|---|---|---|---|---|
| **Degree** | 96.490 | | | |
| **Weighted Degree** | 362.740 | | | |
| | Diameter | Radius | Avg. Path Length | #. of Shortest Paths |
| **Graph Distance** | 2 | 1 | 1.02535353535354 | 9900 |
| | Density | | | |
| **Graph Density** | 0.975 | | | |
| | Modularity | Modularity with resolution | #. of Communities | |
| **Modularity** | 0.029 | 0.029 | 5 | |
| | Number of Weakly Connected Components | Number of Strongly Connected Components | | |
| **Connected Components** | 1 | 1 | | |
| | Avg. Clustering Coefficient | | | |
| **Clustering Coefficient** | 0.975 | | | |

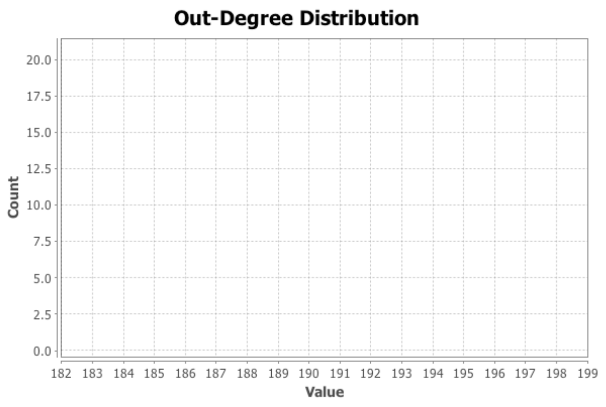
***Table 5.2.5***: *Summary of Network Analysis Reports for the Followers and Enemies Object Relationships of Case 1*

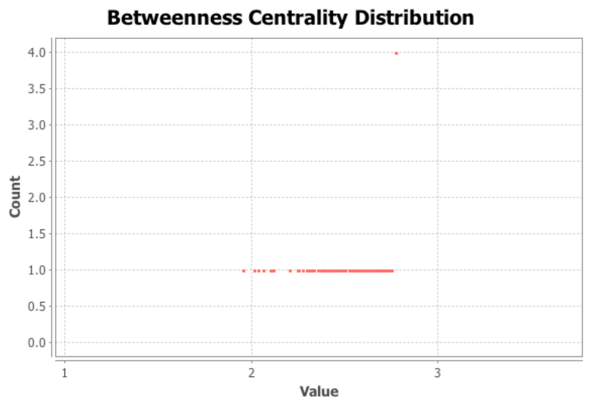
Network analysis of the followers and enemies object relationships immediately reveals useful information about the network. A complete contrast of the ownership and domain relationships, as witnessed by the visualization of the graph, this relationship comprises of much more edges, making it similar to the usage/interaction relationship. An average degree close to 100 shows us that VEs are immediately connected to a vast amount of VEs, making the relationship **generic**. This is the result of the sociability of each VE. As each VE has a commendable amount of “friends” and “enemies”, this way of socializing among VEs leads to a more complete network. The high density of the network of followers and enemies relationships is also an indication of a **dense** network.

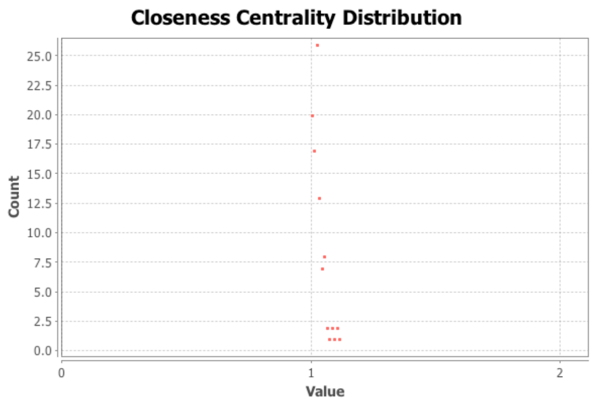
The difference between this relationship and relationships such as the ownership and domain is also evident in the visualizations of the networks. While communities are easily spotted in the latter, betraying the number of users or domains, the former relationship gives no such clues. On the contrary, the density of the network is reinforced by the visualization of the network.

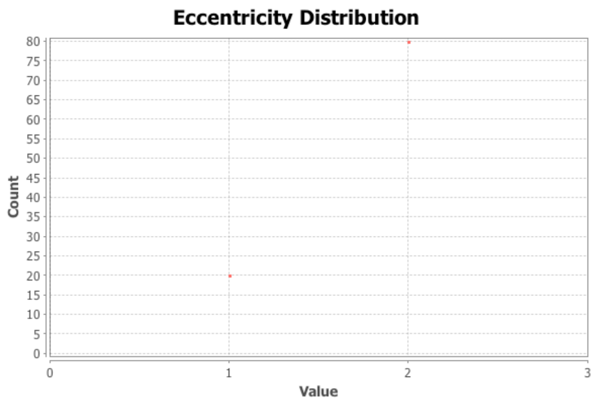

Degree Distribution
Count
Value

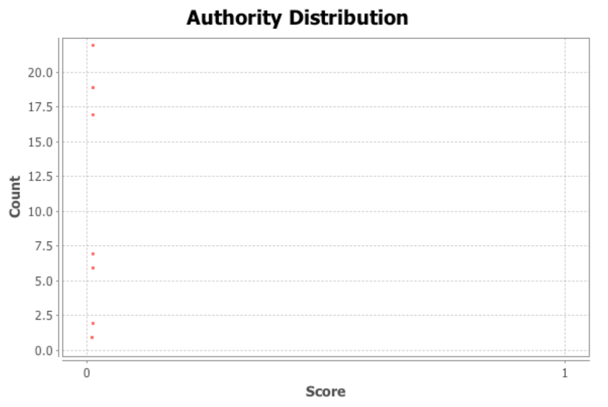

In-Degree Distribution
Count
Value

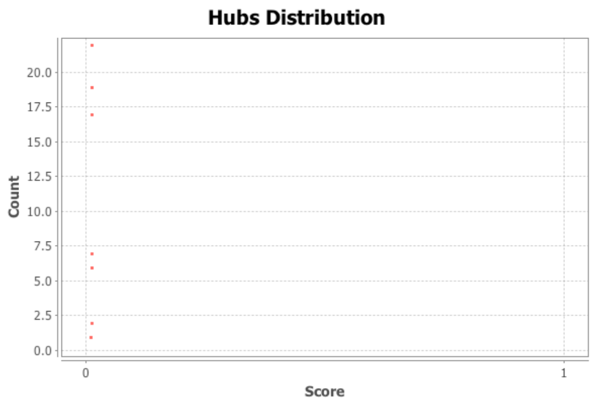

Out-Degree Distribution
Count
Value

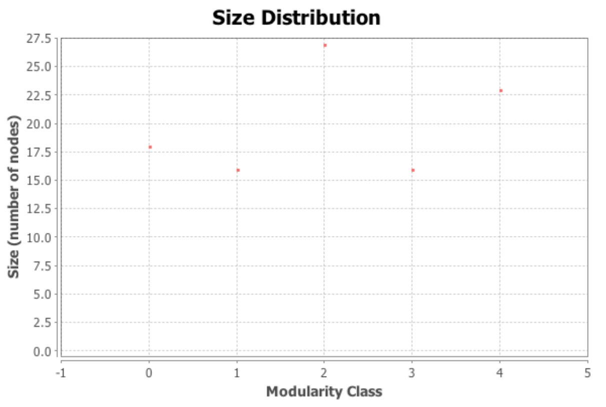

Betweenness Centrality Distribution
Count
Value


Closeness Centrality Distribution
Count
Value


Eccentricity Distribution
Count
Value


Authority Distribution
Count
Score


Hubs Distribution
Count
Score


Size Distribution
Size (number of nodes)
Modularity Class

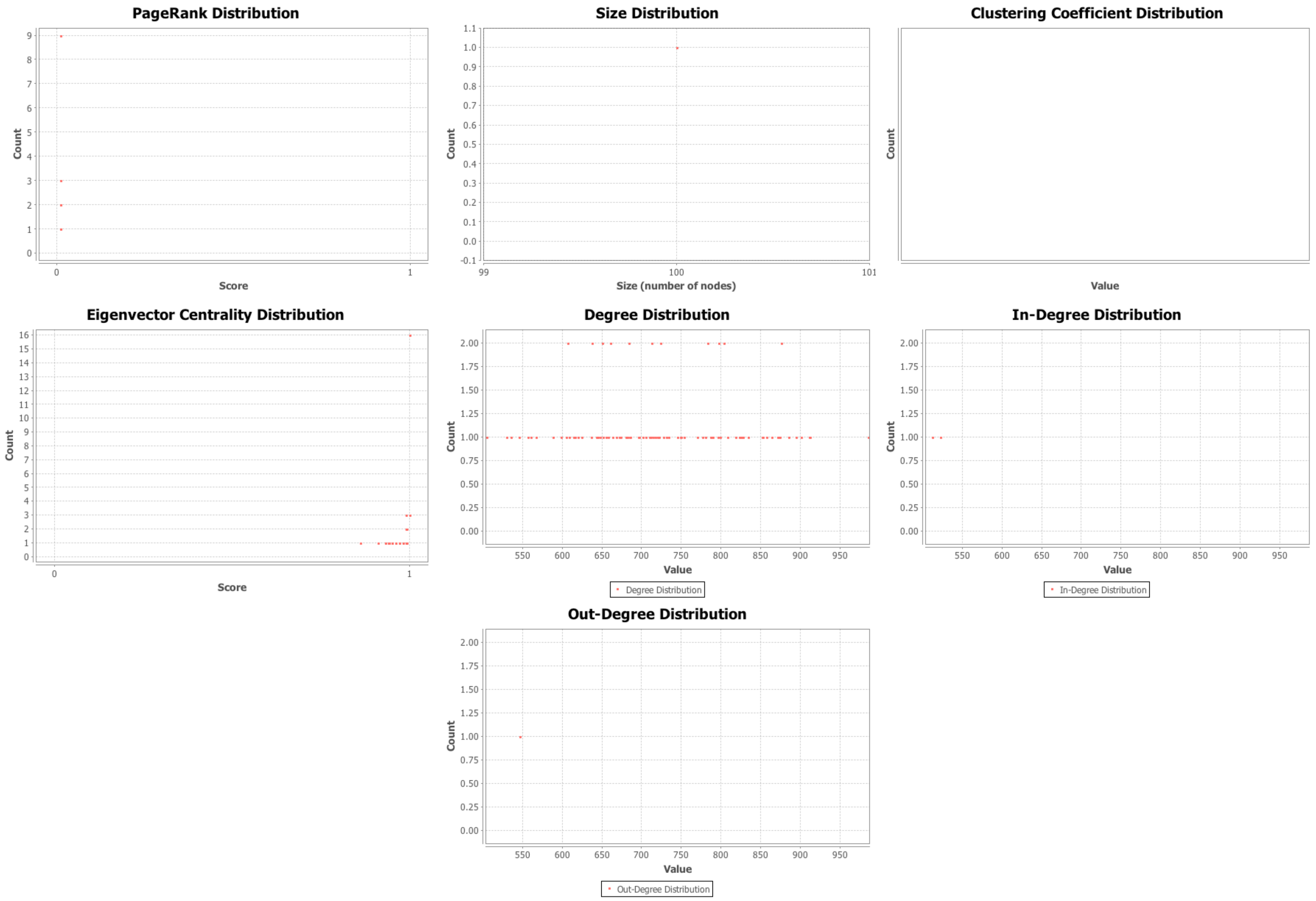


***Figure**: Network Analysis for Followees and Enemies Object Relationships of Case1*

*From top to bottom: Degree Distribution, In Degree Distribution, Out Degree Distribution, Betweenness Centrality Distribution, Closeness Centrality Distribution, Eccentricity Distribution,*
*Authority Distribution, Hubs Distribution, Modularity, PageRank Distribution, Connected Components Distribution, Clustering Coefficient Distribution, Eigenvector Centrality Distribution, Weighted Degree Distribution, In Weighted Degree Distribution, Out Weighted Degree Distribution*

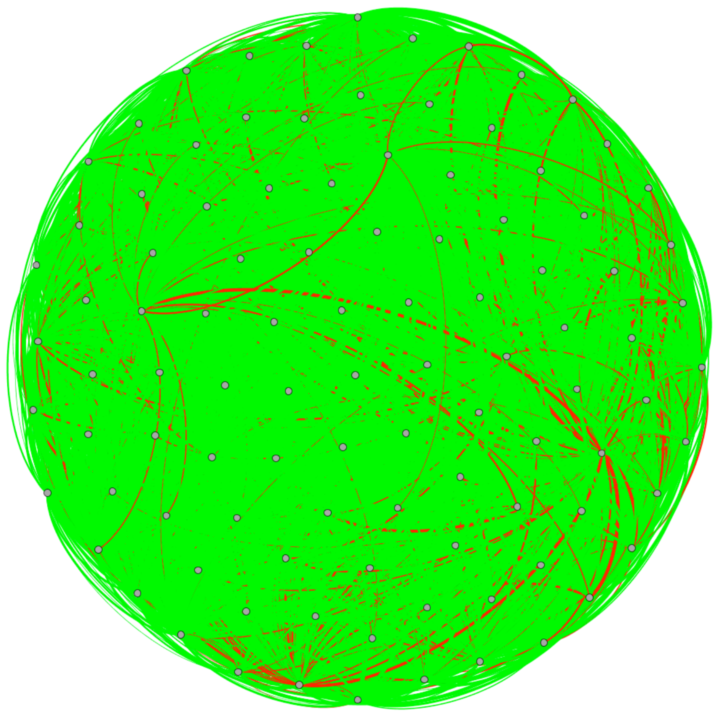


***Figure**: Followees and Enemies Object Relationships Graph*

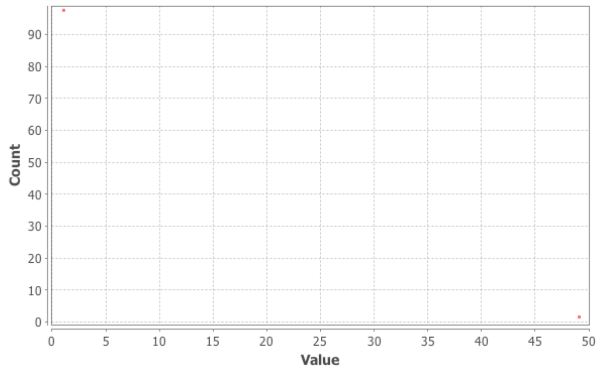
### 5.2.6 Parental Object Relationship

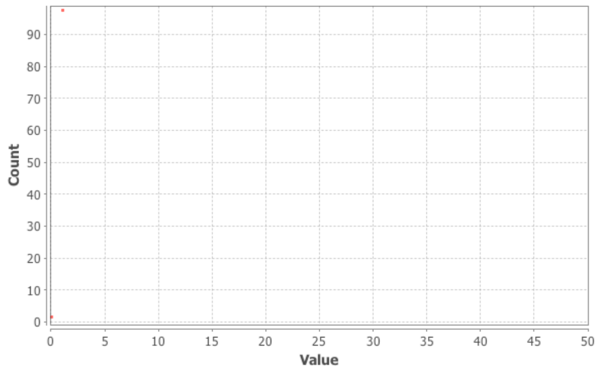
| **Reports** | **Average Degree** | | | |
|---|---|---|---|---|
| **Degree** | 0.980 | | | |
| | Diameter | Radius | Avg. Path Length | #. of Shortest Paths |
| **Graph Distance** | 1 | 0 | 1.0 | 98 |
| | Density | | | |
| **Graph Density** | 0.010 | | | |
| | Modularity | Modularity with resolution | #. of Communities | |
| **Modularity** | 0.500 | 0.500 | 2 | |
| | Number of Weakly Connected Components | Number of Strongly Connected Components | | |
| **Connected Components** | 2 | 100 | | |
| | Avg. Clustering Coefficient | | | |
| **Clustering Coefficient** | 0.000 | | | |

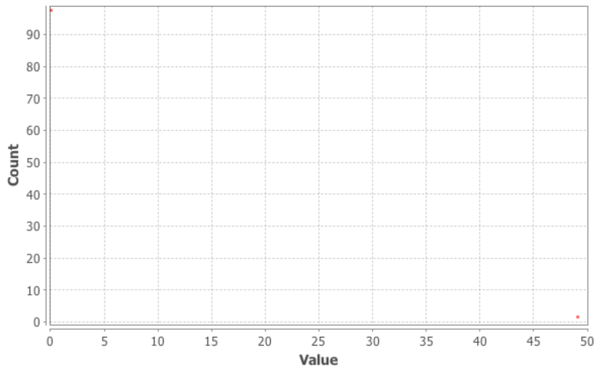
***Table 5.2.6***: *Summary of Network Analysis Reports for Parental Object Relationships of Case 1*

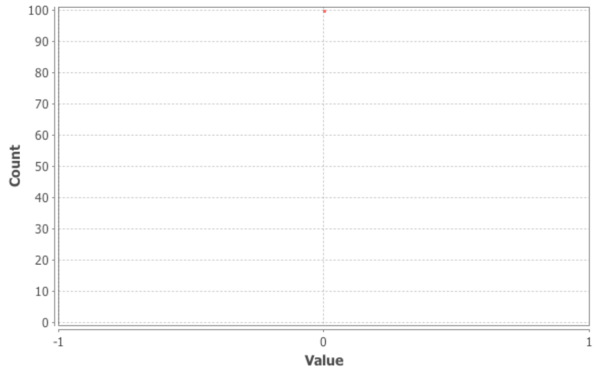
Network analysis of the parental object relationships immediately reveals useful information about the network. An average degree below 1 shows us that VEs are immediately connected to at most one VE, making the relationship very **specific**. This means that using this relationship is preferred when searching a particular VE. The low density of the network of ownership relationships is also an indication of a **sparse** network.

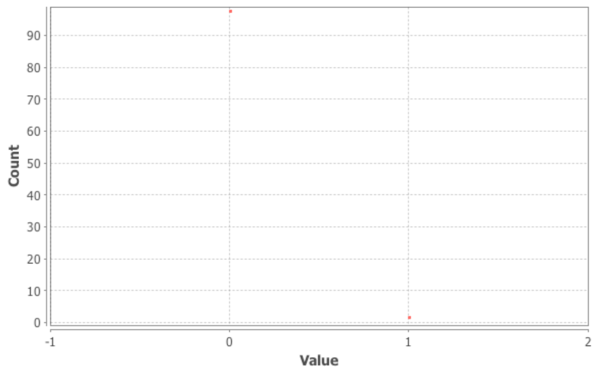
The most interesting piece of information that can be gathered both by the metrics reports and the visualization of the network, regards the number of communities in the network. Both the number of the communities and the number of weakly connected components can lead us to the **number of Head VEs** of the network of Virtual Entities. Each community represents a group of VEs which share the same leader. The sum of these groups is of course the total number of Head VEs in the network. This can also be verified by the statistics of the network of Case1, which is 2 total Head VEs, one for each application, same as the number of communities. The existence of one Head VE per application can also lead us to the **total number of applications** in the network, which is also 2 in this case.

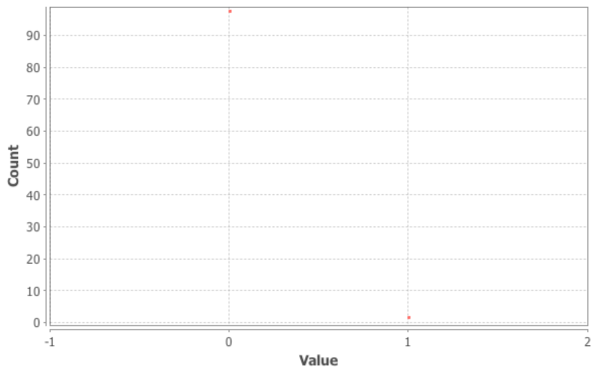

Degree Distribution

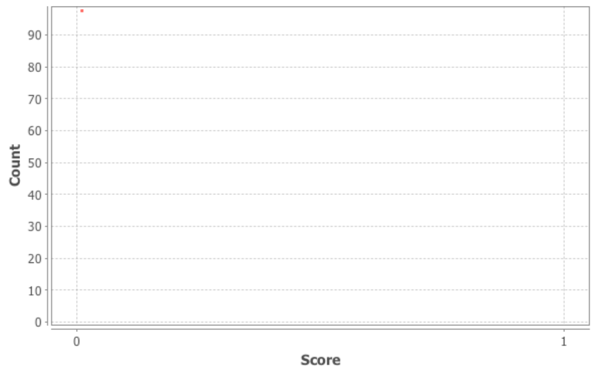

In-Degree Distribution

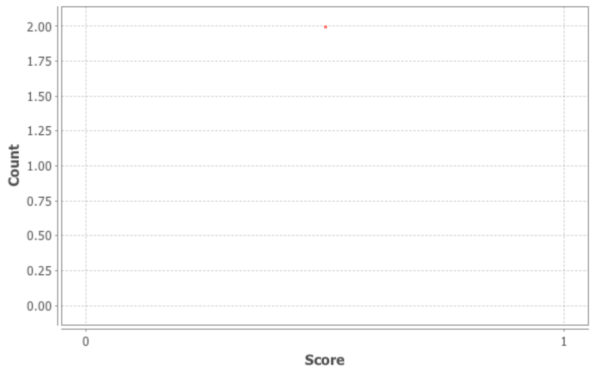

Out-Degree Distribution

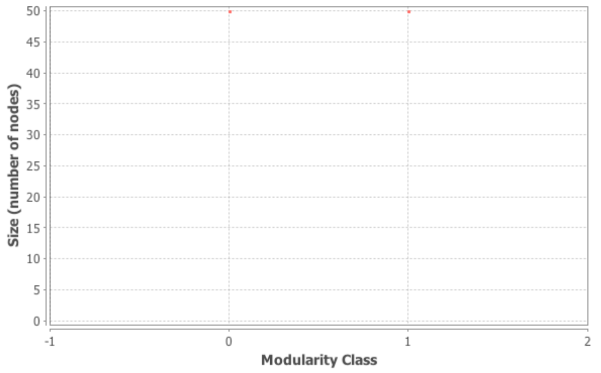

Betweenness Centrality Distribution


Closeness Centrality Distribution


Eccentricity Distribution


Authority Distribution


Hubs Distribution


Size Distribution

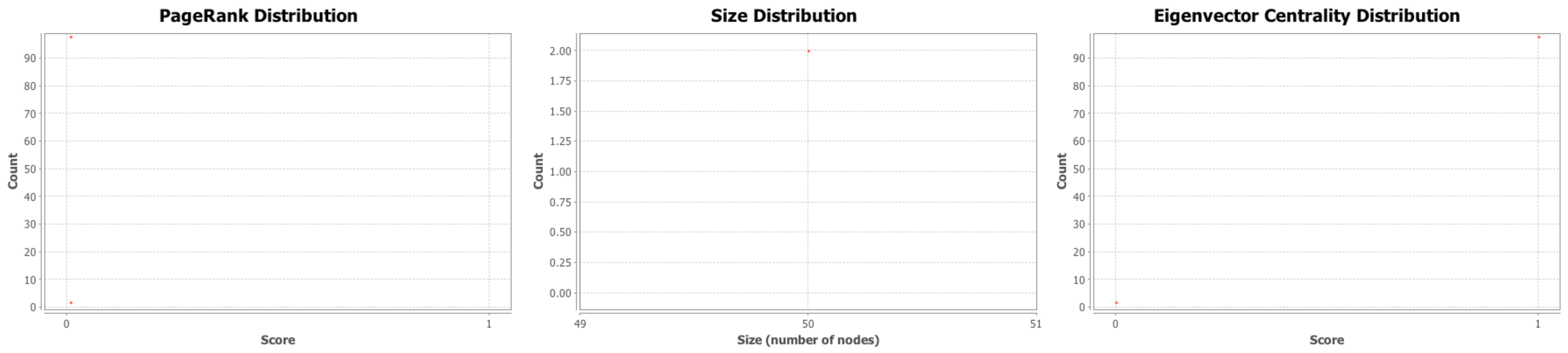


***Figure***: *Network Analysis for Parental Object Relationships of Case1*
*From top to bottom: Degree Distribution, In Degree Distribution, Out Degree Distribution, Betweenness Centrality Distribution, Closeness Centrality Distribution, Eccentricity Distribution,*
*Authority Distribution, Hubs Distribution, Modularity, PageRank Distribution, Connected Components Distribution, Clustering Coefficient Distribution, Eigenvector Centrality Distribution*

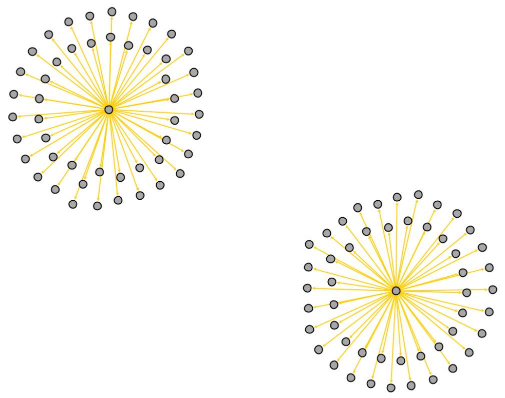


***Figure***: *Parental Object Relationship Graph*

### 5.2.7 Co - Work Object Relationship

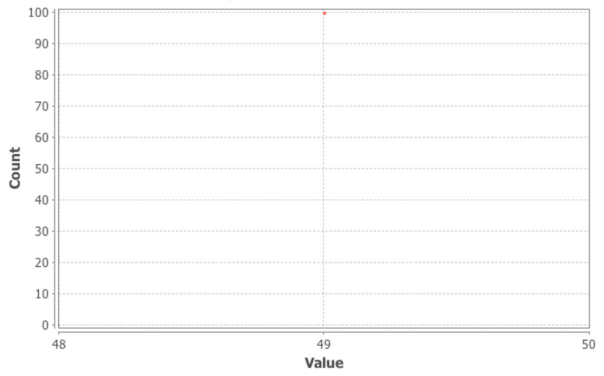
| **Reports** | **Average Degree** | | | |
|---|---|---|---|---|
| **Degree** | 49.000 | | | |
| | Diameter | Radius | Avg. Path Length | #. of Shortest Paths |
| **Graph Distance** | 1 | 1 | 1.0 | 4900 |
| | Density | | | |
| **Graph Density** | 0.495 | | | |
| | Modularity | Modularity with resolution | #. of Communities | |
| **Modularity** | 0.500 | 0.500 | 2 | |
| | Number of Weakly Connected Components | | | |
| **Connected Components** | 2 | | | |
| | Avg. Clustering Coefficient | Total Triangles | | |
| **Clustering Coefficient** | 1.000 | 39200 | | |

***Table 5.2.7***: *Summary of Network Analysis Reports for Co - Work Object Relationships of Case 1*

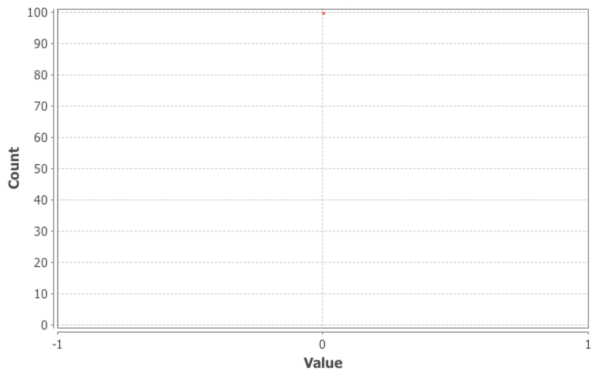
Network analysis of the Co - Work object relationships immediately reveals useful information about the network. An average degree of 50 shows us that VEs are immediately connected to half of the network, making the relationship **generic**. This means that using this relationship is preferred when searching a particular VE. The medium density of the network of co - work relationships is also an indication of a **half dense** network.

The most interesting piece of information that can be gathered both by the metrics reports and the visualization of the network, regards the number of communities in the network. Both the number of the communities and the number of weakly connected components can lead us to the **number of applications** of the network of Virtual Entities. Each community represents a group of VEs which share the same application. The sum of these groups is of course the total number of applications in the network. This can also be verified by the statistics of the network of Case1, which is 2 applications.

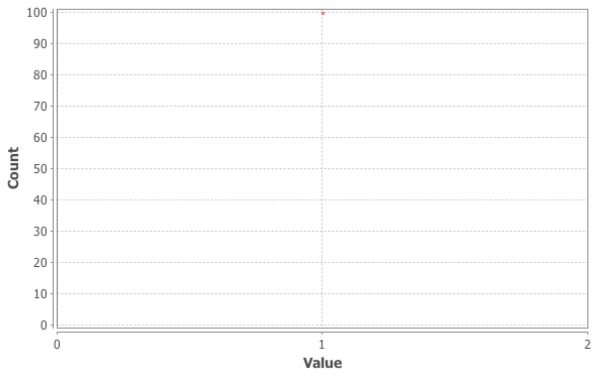
Degree Distribution



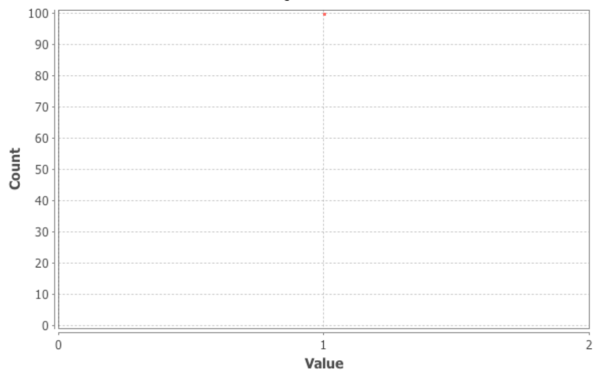
Betweenness Centrality Distribution



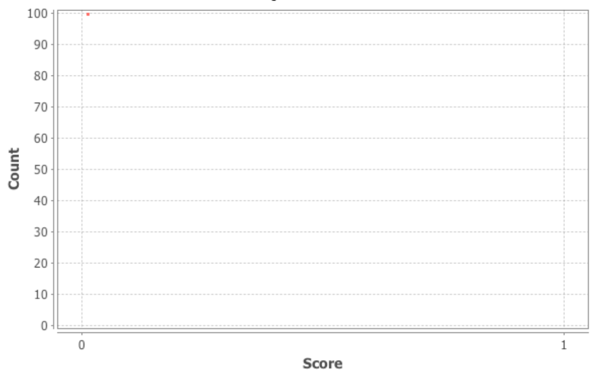
Closeness Centrality Distribution



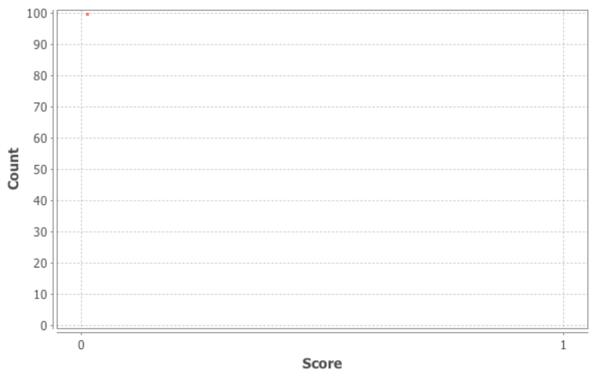
Eccentricity Distribution



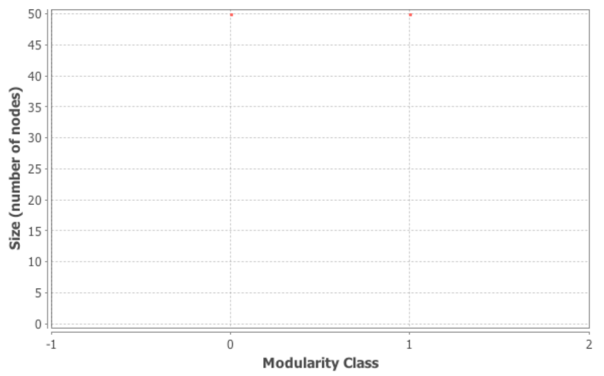
Authority Distribution



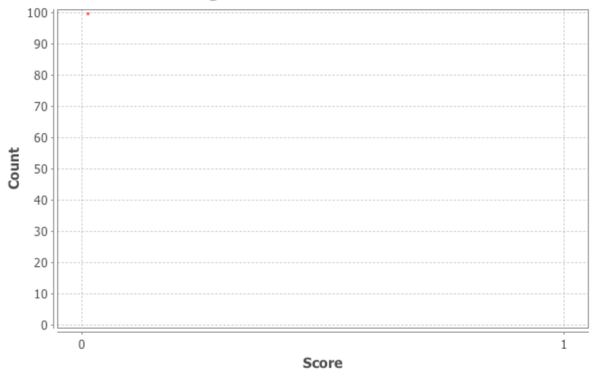
Hubs Distribution



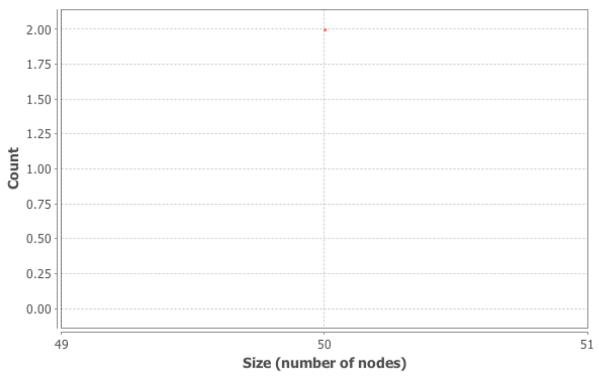
Size Distribution

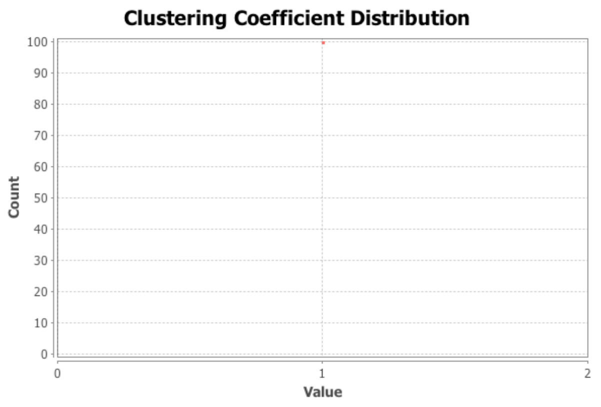


PageRank Distribution



Size Distribution





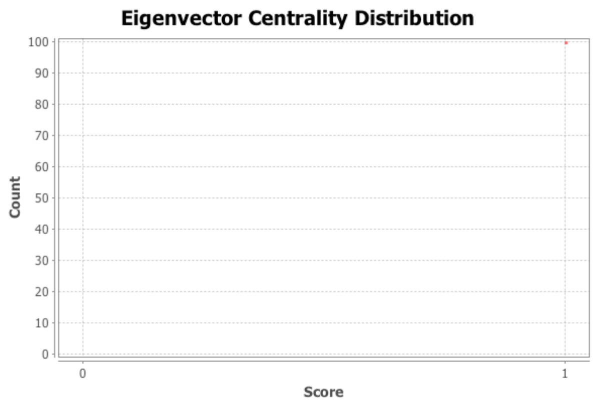


***Figure***: *Network Analysis for Co - Work Object Relationships of Case1*
*From top to bottom: Degree Distribution, Betweenness Centrality Distribution, Closeness Centrality Distribution, Eccentricity Distribution, Authority Distribution, Hubs Distribution, Modularity, PageRank Distribution, Connected Components Distribution, Clustering Coefficient Distribution, Eigenvector Centrality Distribution*

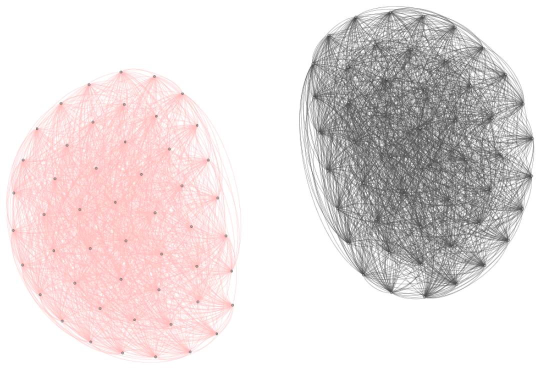


***Figure***: *Co - Work Object Relationship Graph*

### 5.2.8 Conflict Object Relationship

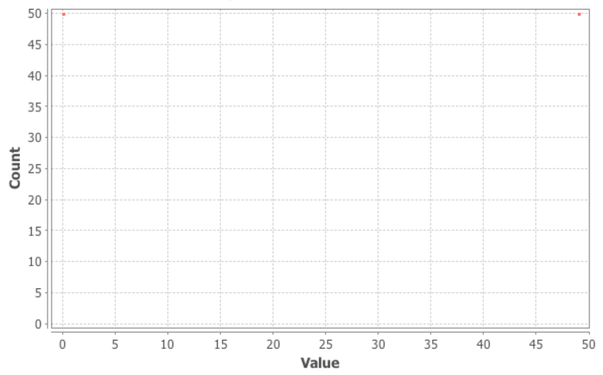
| **Reports** | **Average Degree** | | | |
|---|---|---|---|---|
| **Degree** | 24.500 | | | |
| | Diameter | Radius | Avg. Path Length | #. of Shortest Paths |
| **Graph Distance** | 1 | 0 | 1.0 | 2450 |
| | Density | | | |
| **Graph Density** | 0.247 | | | |
| | Modularity | Modularity with resolution | #. of Communities | |
| **Modularity** | 0.000 | 0.000 | 51 | |
| | Number of Weakly Connected Components | | | |
| **Connected Components** | 51 | | | |
| | Avg. Clustering Coefficient | Total Triangles | | |
| **Clustering Coefficient** | 1.000 | 19600 | | |

***Table 5.2.8****: Summary of Network Analysis Reports for Conflict Object Relationships of Case 1*

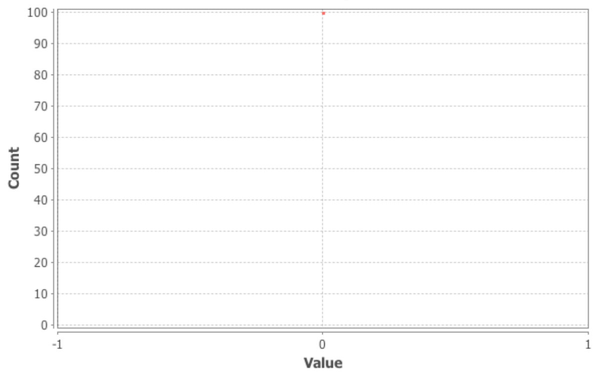
Network analysis of the conflict object relationships rely heavily on the amount of conflict among VEs. In this case, conflict exists only in the first application, which translates to half of the VEs of the network (50). The other 50 VEs do not form any conflict relationships. This affects the average degree greatly. However the low density can provide an image of how **sparse** the network is.

The most interesting piece of information that can be gathered both by the metrics reports and the visualization of the network, regards the number of communities in the network. The inclusion of the non connected nodes of the network give us a total number of communities equal to 51. That is however expected, as each non connected node forms a community. The 51st community, however, is actually the community that betrays the **group of conflicts** in the network. This is strongly witnessed in the graph. As only one application has conflicts, this corresponds to the community that is formed of the conflicting VEs.

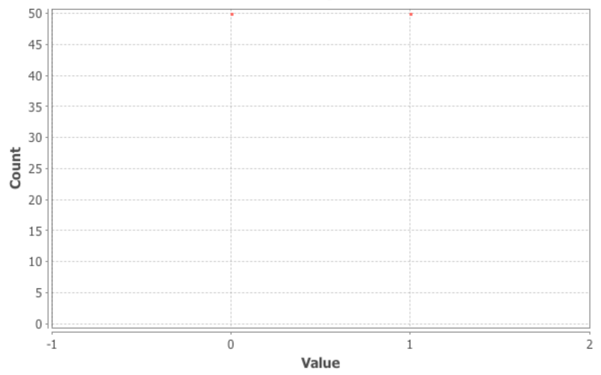
Degree Distribution



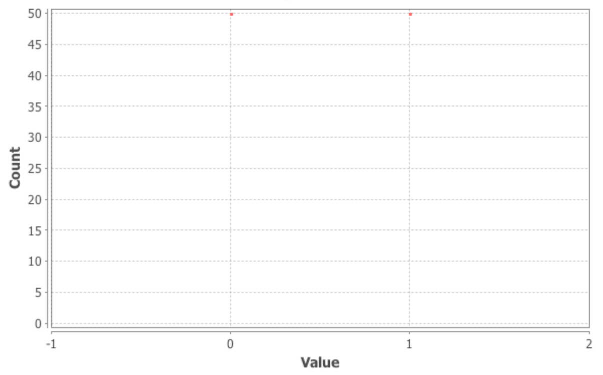
Betweenness Centrality Distribution



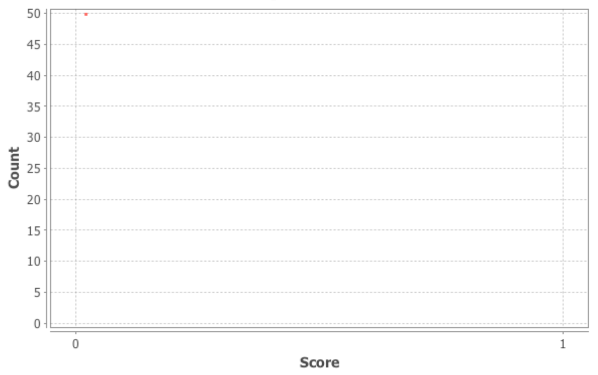
Closeness Centrality Distribution



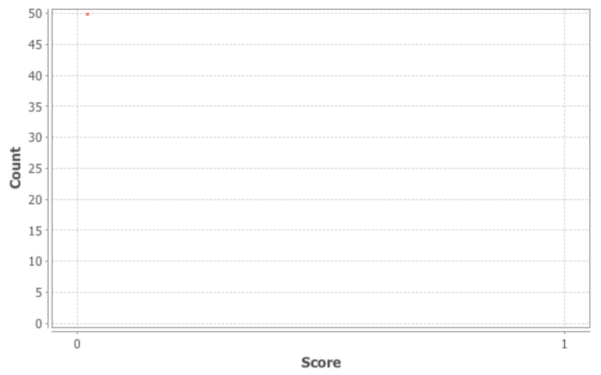
Eccentricity Distribution



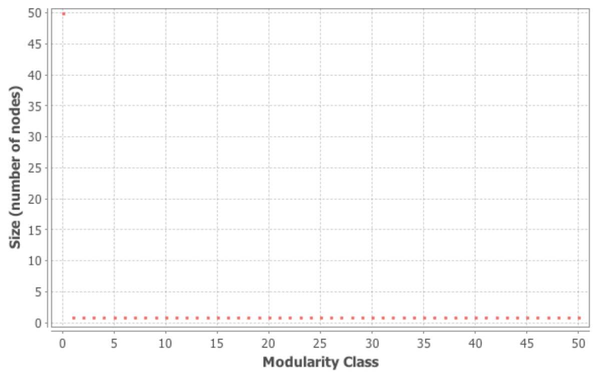
Authority Distribution



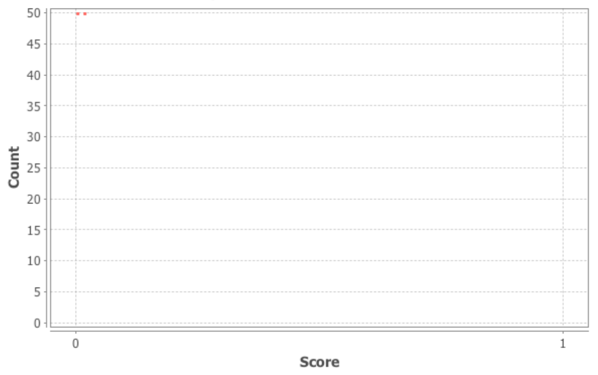
Hubs Distribution



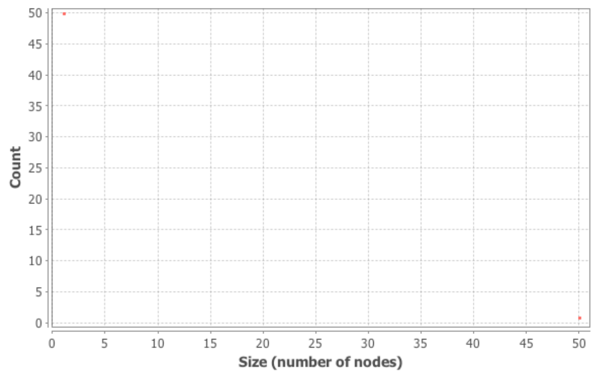
Size Distribution

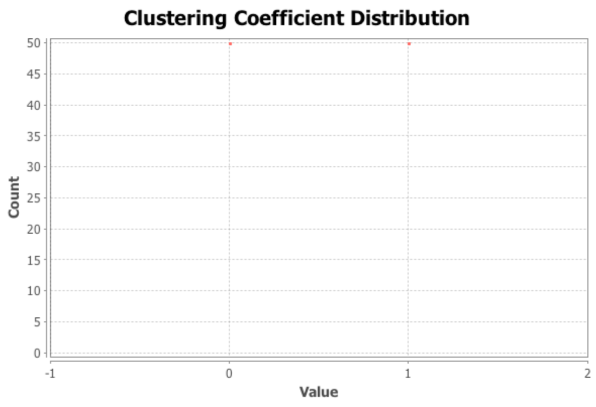


PageRank Distribution



Size Distribution





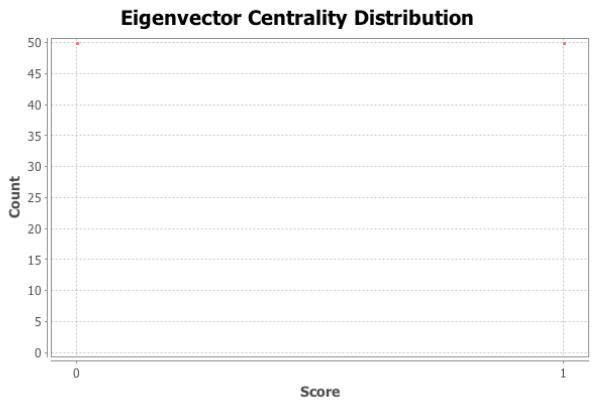


***Figure***: *Network Analysis for Conflict Object Relationships of Case1*
*From top to bottom: Degree Distribution, Betweenness Centrality Distribution, Closeness Centrality Distribution, Eccentricity Distribution, Authority Distribution, Hubs Distribution, Modularity, PageRank Distribution, Connected Components Distribution, Clustering Coefficient Distribution, Eigenvector Centrality Distribution*

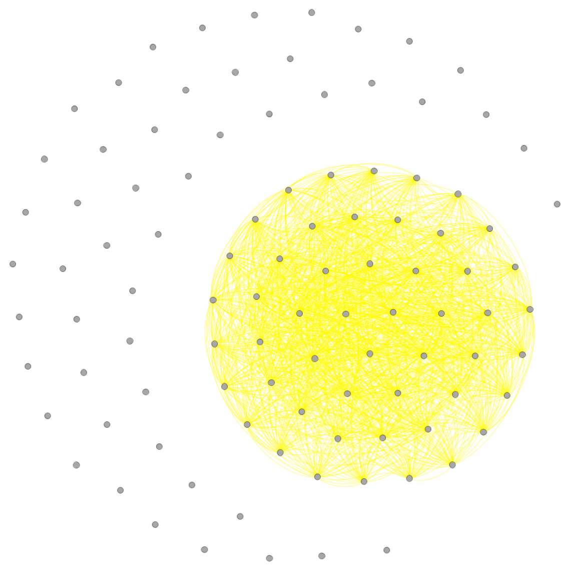


***Figure***: *Conflict Object Relationship Graph*

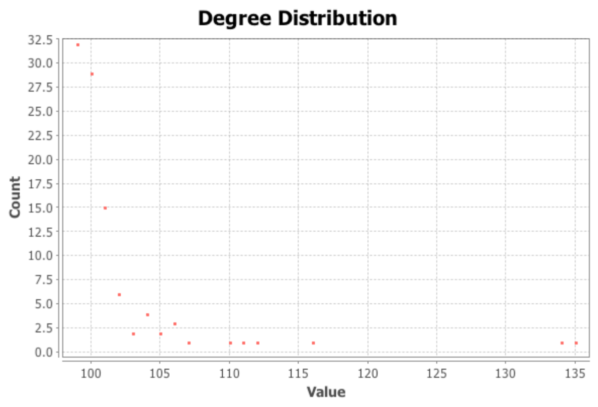
### 5.2.9 Case 1 Complete Network of Object Relationships

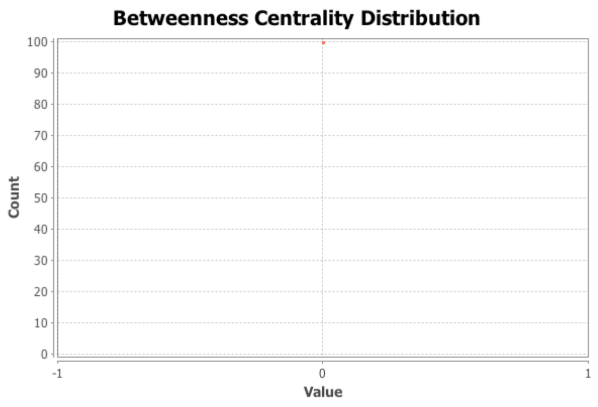
| Reports | Average Degree | | | |
|---|---|---|---|---|
| **Degree** | 101.700 | | | |
| **Weighted Degree** | 407.500 | | | |
| | Diameter | Radius | Avg. Path Length | #. of Shortest Paths |
| **Graph Distance** | 1 | 1 | 1.0 | 9900 |
| | Density | | | |
| **Graph Density** | 1.000 | | | |
| | Modularity | Modularity with resolution | #. of Communities | |
| **Modularity** | 0.043 | 0.043 | 5 | |
| | Number of Weakly Connected Components | | | |
| **Connected Components** | 1 | | | |
| | Avg. Clustering Coefficient | Total Triangles | | |
| **Clustering Coefficient** | 1.000 | 161700 | | |

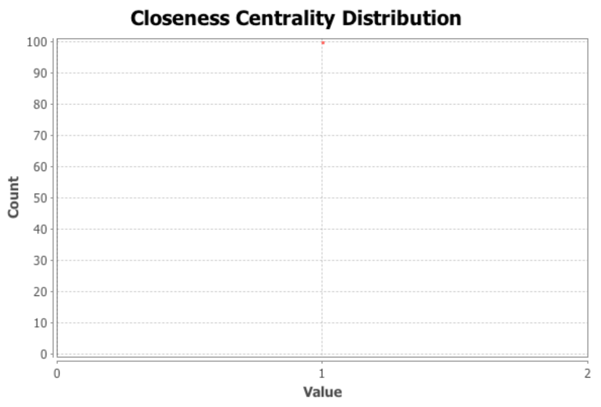
***Table 5.2.9***: *Summary of Network Analysis Reports for the Complete Network of Object Relationships*
*of Case 1*

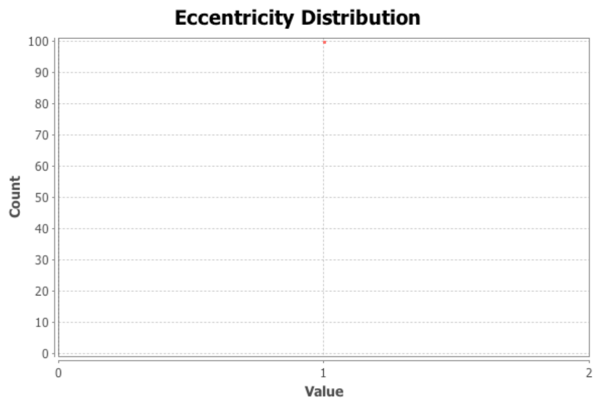
Putting all the previous relationships together creates an abundance of edges in the network. The combination of all object relationships gives a complete network, where all VEs are immediately connected to the rest of the network, in one way or another. The resulting network is **complete**, as evidenced by each density, which proves that it is possible to reach the total amount of VEs by using one or more of the proposed relationships.

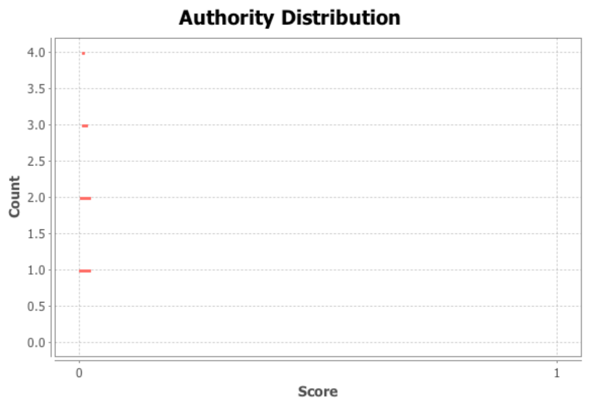
The visualization of the graph makes separating any information about the relationships or the network nearly impossible, due to the density caused by the amount of edges. As mentioned before, Gephi does not support parallel edges and converts them instead to a single edge with weight equal to the sum of actual edges. This is easily noticed in the graph, which appears to be mainly dominated by followers/followees and domain relationships, which is understandable, given the high volume of these relationships and the removal of all edges that connect same pairs of VEs.

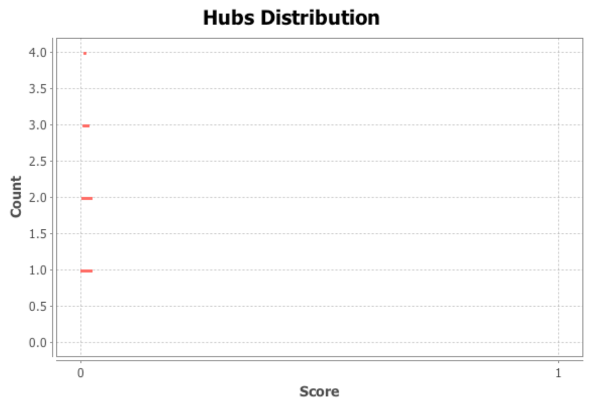

Degree Distribution
Count
Value

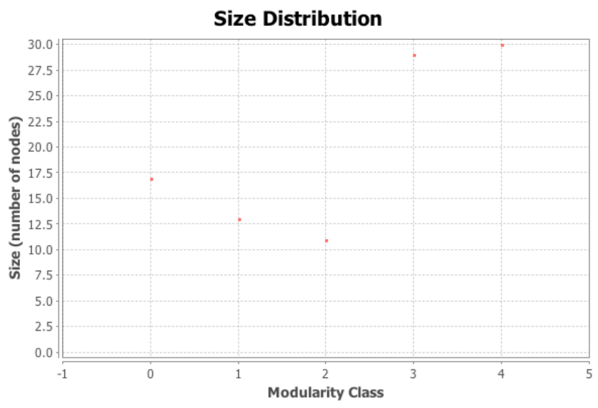

Betweenness Centrality Distribution
Count
Value

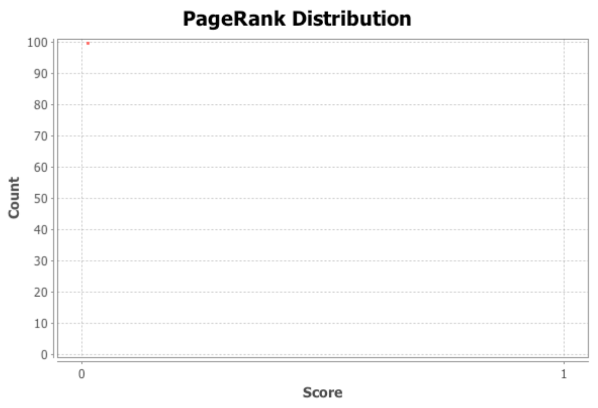

Closeness Centrality Distribution
Count
Value

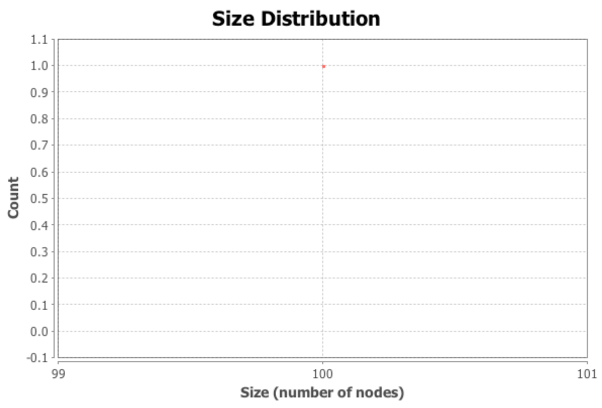

Eccentricity Distribution
Count
Value


Authority Distribution
Count
Score


Hubs Distribution
Count
Score


Size Distribution
Size (number of nodes)
Modularity Class


PageRank Distribution
Count
Score


Size Distribution
Count
Size (number of nodes)

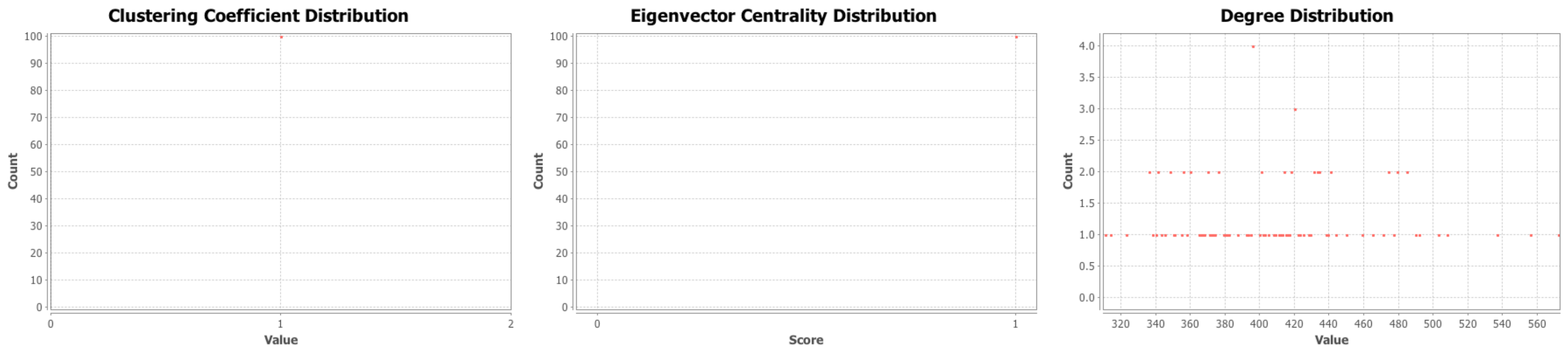


***Figure***: *Network Analysis for the Complete Network of Object Relationships of Case1*
*From top to bottom: Degree Distribution, Betweenness Centrality Distribution, Closeness Centrality Distribution, Eccentricity Distribution, Authority Distribution, Hubs Distribution, Modularity, PageRank Distribution, Connected Components Distribution, Clustering Coefficient Distribution, Eigenvector Centrality Distribution, Weighted Degree Distribution*

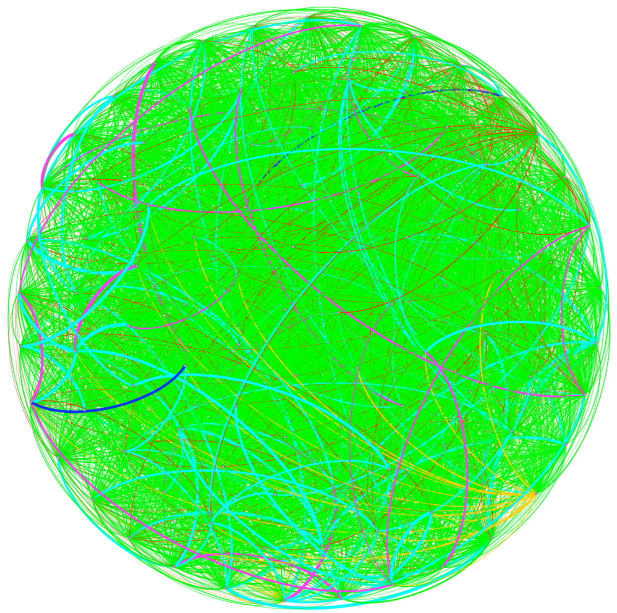


***Figure***: *Complete Network of Object Relationships Graph*

## 5.3 Case 2: 500 Virtual Entities

The following table summarizes the basic statistics for Case 2:

| Case 2 | |
|---|---|
| **Total Nodes** | 500 |
| **Total Users** | 125 |
| **Domains** | 8 |
| **Applications** | 2 |
| **Max Friends/Enemies** | 300 |
| **Ownership Relationships** | 1057 |
| **Usage/Interaction Relationships** | 1049023 |
| **Domain Relationships** | 15530 |
| **Followers Relationships** | 4919532 |
| **Enemies Relationships** | 2767881 |
| **Parental Relationships** | 498 |
| **Co - Work Relationships** | 62250 |
| **Conflict Relationships** | 31125 |
| **Total Relationships** | **8846896** |

***Table* 5.3**: *Case 2 Statistics*

The visualization of the network for each object relationship is given in the following figures:

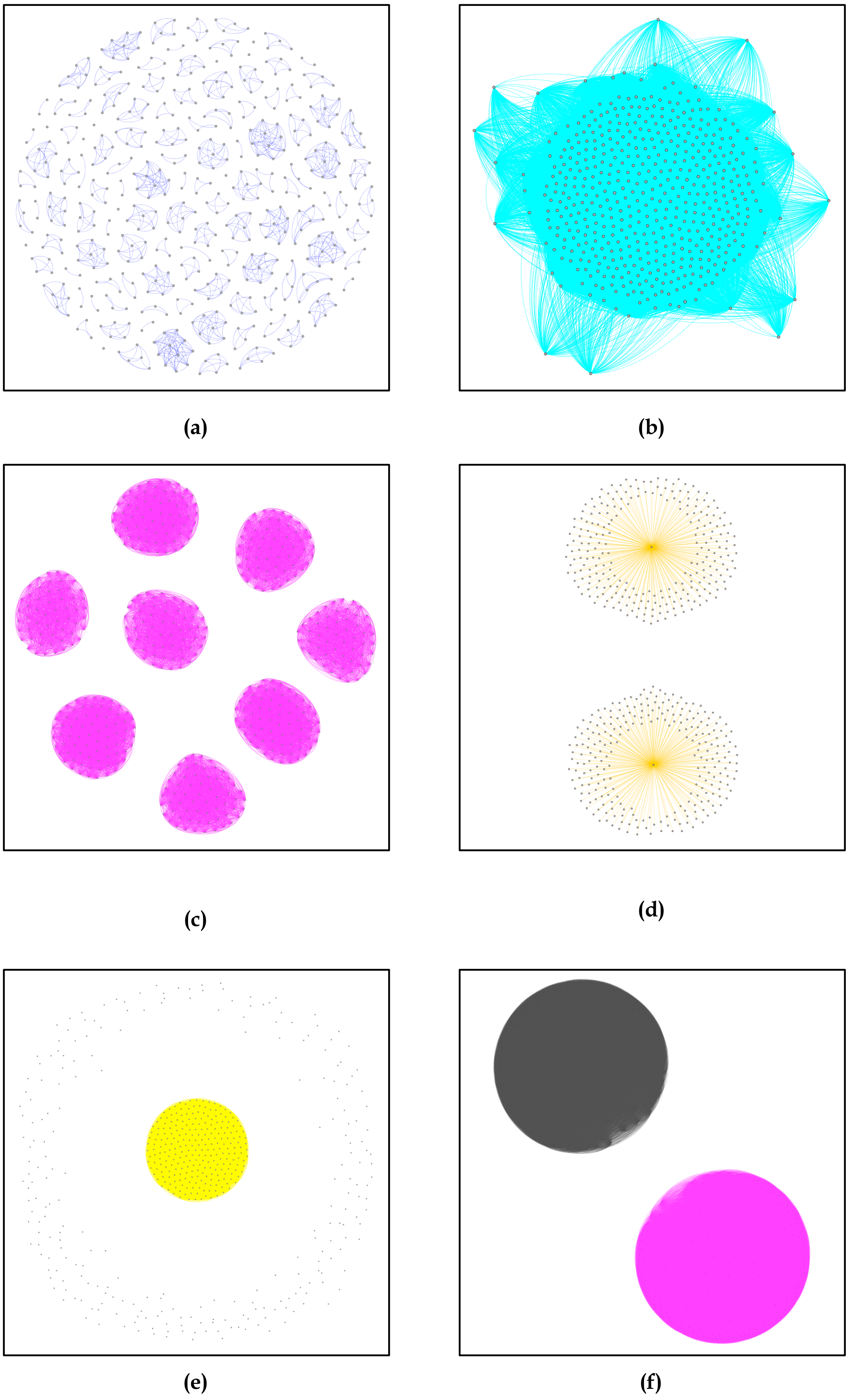


***Figure 5.3***: *Object Relationship Networks for Case 2:* ***(a)*** *Ownership,* ***(b)*** *Usage,* ***(c)*** *Domain,* ***(d)*** *Parental,* ***(e)*** *Conflict and* ***(f)*** *Co Work Object Relationships*

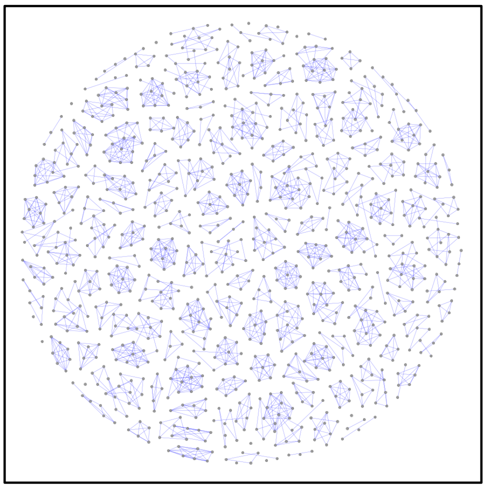
## 5.4 Case 3: 1000 Virtual Entities

The following table summarizes the basic statistics for Case 3:

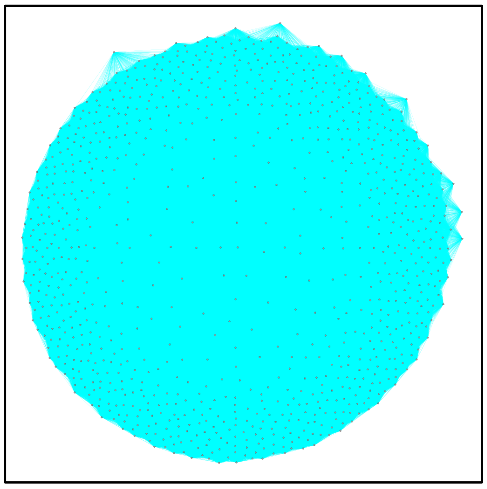
| Case 3 | |
|---|---|
| **Total Nodes** | 1000 |
| **Total Users** | 250 |
| **Domains** | 8 |
| **Applications** | 2 |
| **Max Friends/Enemies** | 600 |
| **Ownership Relationships** | 2025 |
| **Usage/Interaction Relationships** | 7970416 |
| **Domain Relationships** | 62211 |
| **Followers Relationships** | 37348915 |
| **Enemies Relationships** | 23149683 |
| **Parental Relationships** | 998 |
| **Co - Work Relationships** | 249500 |
| **Conflict Relationships** | 124750 |
| **Total Relationships** | **68908498** |

***Table 5.4***: *Case 3 Statistics*

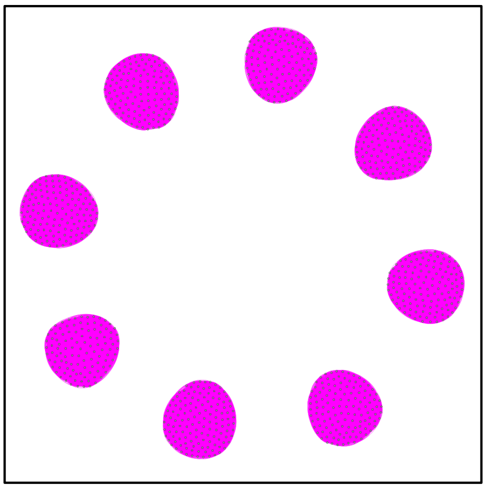
The visualization of the network for each object relationship is given in the following figures:



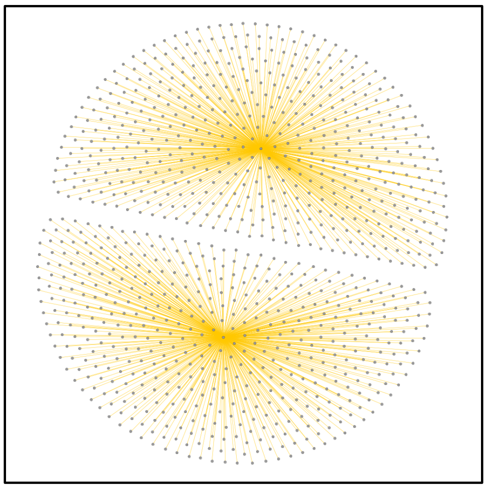
**(a)**



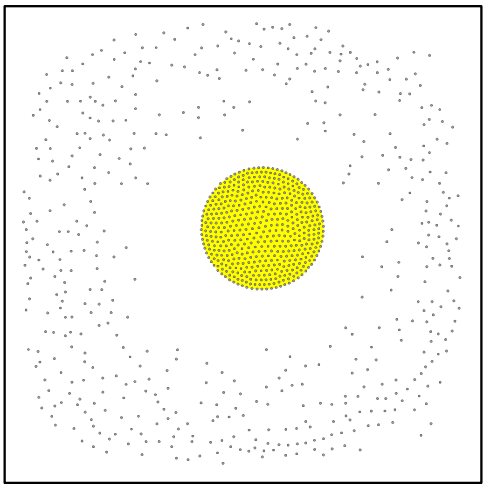
**(b)**



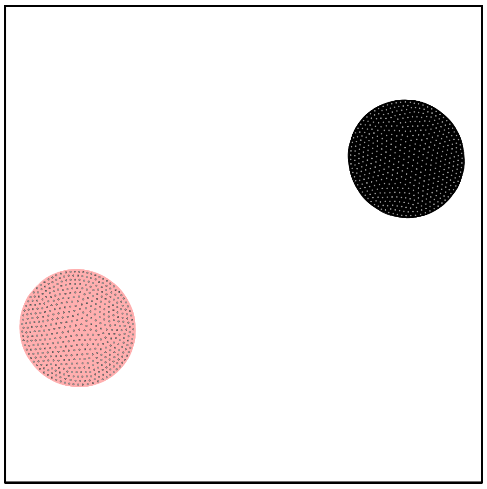
**(c)**



**(d)**



**(e)**



**(f)**

***Figure 5.4***: *Object Relationship Networks for Case 3:* ***(a)*** *Ownership,* ***(b)*** *Usage,* ***(c)*** *Domain,* ***(d)*** *Parental,* ***(e)*** *Conflict and* ***(f)*** *Co Work Object Relationships*

## 5.5 Data Generation

As COSMOS is still in its infancy, an efficient dataset needed to be created for network analysis. To this end, we used **Python** scripts to generate the network of VEs as well as application and VE characteristics. Then, based on these characteristics, the object relationships among VEs are gathered and formatted properly into GraphML lines of code, which are then exported for use in Gephi. This chapter contains code snippets, along with detailed comments.

### 5.5.1 Network and Application Characteristics

```
##Number of VEs and Number of Users:
Net_size =  int(sys.argv[1]);
Users = math.floor(0.25*Net_size);
## Domains:
Domains = ['Smart Cities','Smart Water', 'eHealth', 'Retail', 'Logistics',
'Home Automation', 'Security Emergencies', 'Smart Metering'];
## Max number of friends, enemies and users each VE can have:
max_fren = math.floor(0.6*Net_size);
max_users = math.floor(0.6*Users);

## Printing Statistics:
print 'Users: ' + str(Users);
print 'Frienemies: ' + str(max_fren);

## 2 Applications in the Network.
## Each Application will be assigned to one application only

## APPLICATION 1: Net_size/2 first VEs in App1, random Head VE among them.
App1 = {};
App1['AppID'] = 'A1';
App1['VEs'] = [];
App1['HeadVE'] =  randrange(1, Net_size/2 +1,1);

for i in range(1, Net_size/2 +1):
    App1['VEs'].append('VE' + str(i));


## APPLICATION 2: Remaining VEs in App2

App2 = {};
App2['AppID'] = 'A2';
App2['VEs'] = [];
for i in range(Net_size/2 +1, Net_size +1,1 ):
    App2['VEs'].append('VE' + str(i));

App2['HeadVE'] =  randrange(Net_size/2 +1, Net_size +1 ,1);
```

*Code 5.5.1: Setting basic Network and Application info*

In this section, the Network and Application characteristics are set. The number of nodes is read from input arguments, the number of users is set to 25% of the number of nodes, the 8 Domains are gathered in a list and the maximum numbers of Followers, Enemies and Users a VE can have are set to 60%. The first Application contains the first half of VEs and a Head VE a random VE from this set and the second Application contains the second half of VEs and a Head VE a random VE from this set.

### 5.5.2 Ownership Object Relationship

```
OwnershipList = {}; ## List containing a list of VEs for each Owner

## Function to assign owners to each VE
def assign_owners():

    for j in range(1,Net_size+1):

      ## Generate a random owner:
        owner = 'U' + str(randrange (1, Users + 1, 1));
        ## Sort the VE into the list of its owner:
        if owner in OwnershipList:
            ve = 'VE'+ str(j);
            if ve not in OwnershipList[owner]:
                OwnershipList[owner].append(ve);
        else:
                OwnershipList[owner] = [];
                OwnershipList[owner].append('VE'+ str(j));
    return;


OwnRel = []; ## List containing all Ownership Object Relationships

## Function to create the Ownership Object Relationships
def make_ownership():

      ## Sort VEs into Owners Lists:
    assign_owners();
    for owners in OwnershipList:
      ## For every Owner List, create Relationships with VEs of the List:
        velist = OwnershipList[owners];
        for i in range(0, len(velist)):
            for j in range(i, -1,-1):
                if i != j:
                    rel = velist[i] + '-' + velist[j];
                    OwnRel.append(rel);
    return;

make_ownership();
```

***Code* 5.5.2**: *Ownership Object Relationship*

In this section, the Ownership Object Relationship is presented. First of all, each VE is assigned to an Owner. Then, for each Owner List, that contains the VEs the owner has, an Ownership Object Relationship is created among all VEs of the list. All Ownership Object Relationships are then stored in the OwnRel list.

### 5.5.3 Usage/Interaction Object Relationship

```
UsageList = {}; ## List of Lists of VEs for each User

def assign_users():

    for j in range(1,Net_size+1):
        ## For Each VE, generate its users and sort VE in its User Lists
        num =  randrange(1,max_users+1,1)
        for i in range(num):
            usr = 'U' + str(randrange (1, Users + 1, 1));
            if usr in UsageList:
                ve = 'VE'+ str(j);
                if ve not in UsageList[usr]:
                    UsageList[usr].append(ve);
            else:
                UsageList[usr] = [];
                UsageList[usr].append('VE'+ str(j));
    return


UsageRel = []; ## List that contains all Usage/Interaction Object
Relationships
def make_usage():

    assign_users();
    for users in UsageList:
        ## for each User List, create Usage/Interaction Relationships
among VEs in the list:
        velist = UsageList[users];
        for i in range(0, len(velist)):
            for j in range(i, -1,-1):
                if i != j:
                    rel = velist[i] + '-' + velist[j];
                    UsageRel.append(rel);
    return;

make_usage(); ## UsageRel has the usage relationships
```

**Code 5.5.3**: *Usage/Interaction Object Relationship*

In this section, the Usage/Interaction Object Relationship is presented. First of all, each VE is assigned a number of Users. Then, for each User List, that contains the VEs the user has, a Usage/Interaction Object Relationship is created among all VEs of the list. All Usage/Interaction Object Relationships are then stored in the UsageRel list.

## 5.5.4 Domain Object Relationship

```
DomainList = {}; ## List of Domain Lists. Each Domain List holds the VEs of
the Domain

def assign_domain():

    for i in range(1,Net_size+1):
        ## Assign a Domain to the VE:
        domain = Domains[ randrange(0, len(Domains), 1) ];
        ## Sort the VE into the right Domain List:
        if domain in DomainList:
            DomainList[domain].append('VE'+ str(i));
        else:
            DomainList[domain] = [];
            DomainList[domain].append('VE'+ str(i));

    return DomainList;

DomainRel = [];
def make_domain():

    assign_domain();
    ## For each Domain List, make Domain Object Relationships among VEs in
the List:
    for dom in DomainList:
        domlist = DomainList[dom];
        for i in range(0, len(domlist)):
            for j in range(i, -1,-1):
                if i != j:
                    rel = domlist[i] + '-' + domlist[j];
                    DomainRel.append(rel);
    return;


make_domain(); ## DomainRel has the Domain relationships
```

*Code* 5.5.4: *Domain Object Relationship*

In this section, the Domain Object Relationship is presented. First of all, each VE is assigned a Domain. Then, for each Domain List, that contains the VEs of the domain, a Domain Object Relationship is created among all VEs of the list. All Domain Object Relationships are then stored in the DomainRel list.

### 5.5.5 Followers/Followees and Enemies Object Relationships

```
FollowersList = {}; ## Contains a List for every VE which contains its
Followees
EnemiesList = {}; ## Contains a List for every VE which contains its Enemies

def assign_followees():

    for j in range(1,Net_size+1):
        ## Assign a number of VEs as followers and sort into Followers Lists
        num =  randrange(1,max_fren+1,1);
        ve = 'VE' + str(j);
        for i in range(num):
            new = randrange (1, Net_size + 1, 1);
            friend = 'VE' + str(new);
            if new != j:
                if ve in FollowersList:
                    if friend not in FollowersList[ve]:
                        FollowersList[ve].append(friend);
                else:
                    FollowersList[ve] = [];
                    FollowersList[ve].append(friend);
    return

def assign_enemies():

    for j in range(1,Net_size+1):
        ## Assign a number of VEs as enemies and sort into Enemies Lists
        ## A VE cannot be both a followee and an enemy of another VE
        num =  randrange(1,max_fren+1,1);
        ve = 'VE' + str(j);
        for i in range(num):
            new = randrange (1, Net_size + 1, 1);
            enemy = 'VE' + str(new);
            if new != j and ve in FollowersList and enemy not in
FollowersList[ve]:
                if ve in EnemiesList:
                    if enemy not in EnemiesList[ve]:
                        EnemiesList[ve].append(enemy);
                else:
                    EnemiesList[ve] = [];
                    EnemiesList[ve].append(enemy);
    return
```

***Code 5.5.5.1**: Followers/Followees and Enemies Object Relationships: Making the Lists*

In this section, both the Followers/Followees as well as the Enemies Object Relationships are presented. These relationships need to be calculated together, in order to avoid conflicts when a followee VE is also an enemy of at the same time.
First of all, each VE is assigned a number of followees and each followee is sorted into the FollowersList of the follower VE. Similarly, each VE is then assigned a number of enemies and each enemy is sorted as well into the EnemiesList, simultaneously eliminating the case of a VE being both a followee and an enemy at the same time.

```
FollowersRel = [];
EnemiesRel = [];

def make_friends():

    ## Build the Followers Lists and for every List create Followers/Followees 
Obj. Rels. among VEs
    assign_followees();
    for friends in FollowersList:
        frlist = FollowersList[friends];
        for i in range(0, len(frlist)):
            for j in range(i, -1,-1):
                if i != j:
                    rel = frlist[i] + '-' + frlist[j];
                    FollowersRel.append(rel);
    return;

def make_enemies():

    ## Build the Enemies Lists and for every List create Enemies Obj. Rels. 
among VEs
    assign_enemies();
    for enemies in EnemiesList:
        frlist = EnemiesList[enemies];
        for i in range(0, len(frlist)):
            for j in range(i, -1,-1):
                if i != j:
                    rel = frlist[i] + '-' + frlist[j];
                    EnemiesRel.append(rel);
    return;

make_friends();
make_enemies();
```

***Code* 5.5.5.2**: *Followers/Followees and Enemies Object Relationships: Making the Relationships*

Then, for each Followers List, that contains the followee VEs of the follower, a Follower/Followee Object Relationship is created among all VEs of the list. All Followers/Followees Object Relationships are then stored in the FollowersRel list. Similarly, for each Enemies List, that contains the enemy VEs of the VE, an Enemy Object Relationship is created among all VEs of the list. All Enemies Object Relationships are then stored in the EnemiesRel list.

### 5.5.6 Parental Object Relationship

```
Parental = []; ## List of Parental Object Relationships
def make_parental (lower, upper, head):

    ## For each VE of the application, create a Parental Obj. Re. with
the Head Ve of the App.
    for i in range (lower, upper +1):
        if i != head :
            rel = 'VE'+ str(head) + '-' + 'VE' + str(i);
            Parental.append(rel);


make_parental(1,Net_size/2, App1['HeadVE']);
make_parental(Net_size/2 +1,Net_size, App2['HeadVE']); ## Parental has
the parental relationships
```

*Code 5.5.6: Parental Object Relationship*

The Parental Object Relationship is one of the simplest Object Relationships. All that is needed are the VEs and the Head VE of the Application to create the relationships between the Head VE and each VE of the application. This process is made even more trivial by the fact that, as assumed at the beginning, the first half of the VEs belong to Application 1 and the second half to Application 2 and the VEs are serially named. The Parental Object Relationships are stored in the Parental list.

### 5.5.7 Conflict Object Relationship

```
Conflict = []; # List of Conflict Relationships

def make_conflict(lower, upper):

    ## For each VE, make Conflict Obj. Rels with all other VEs
    for i in range(lower,upper+1):
        for j in range(i,0,-1):
            if i != j:
                rel = 'VE'+ str(i) + '-' + 'VE' + str(j);
                Conflict.append(rel);

## Only first Application has Conflicts in our scenarios:
make_conflict(1,Net_size/2);
```

*Code 5.5.7: Conflict Object Relationship*

The Conflict Object Relationship is equally trivial to generate, based on the assumption that in our scenarios conflict exists only for the VEs of Application 1. As all the VEs can be easily retrieved, as mentioned above, it is only a matter of creating the relationships among those VEs. The Conflict Object Relationships are stored in the Conflict list.

### 5.5.8 Co - Work Object Relationship

```
CoWork = []; ## Contains all Co Work Object Relationships

def make_cowork(lower, upper):

    ## Create the relationships among all specified VEs:
    testlist = [];
    for i in range(lower,upper+1):
        for j in range(i,lower-1,-1):
            if i != j:
                rel = 'VE'+ str(i) + '-' + 'VE' + str(j);
                testlist.append(rel);

    CoWork.append(testlist);


## Make the Co - Work Obj. Rels for App 1:
make_cowork(1,Net_size/2);
## Make the Co - Work Obj. Rels for App 2:
make_cowork(Net_size/2 +1,Net_size);
```

***Code 5.5.8**: Co - Work Object Relationship*

Finally, the Co - Work Object Relationships are established in two stages. First, the relationships among VEs of Application 1 are established and then the relationships among VEs of Application 2 follow.

### 5.5.9 Observations and Conclusions

It is made clear that efficient storing of VEs is vital to effortlessly establishing these relationships. Two assumptions are critical to this end: a) VEs are serially named e,g VE1, VE2, VE3 etc. and b) Applications are also serially stored, meaning that the first half of the VEs belong to Application 1 and the second half belong to Application 2. This, in turn, is crucial in reducing the complexity of searching and creating these relationships. However, these assumptions cannot always be true in real life scenarios as well. VEs are subject to arbitrary naming and applications are not usually limited. However, it is important to design an efficient storing  technique of VEs and Applications, that will lead to significant improvements in terms of complexity.

As networks grow bigger and more complex, memory and disk storage requirements are also affected. A significant amount of memory needed to be allocated to Gephi in order to be able to handle our networks, way beyond its default 512MB, i.e a network of 1000 nodes required more than 5GB of memory devoted to Gephi. This, however, is not surprising, as graphical applications such as Gephi are known to be memory intensive. In terms of disk storage, the vast amount of relationships translate to bigger GraphML files. The table below summarizes the space required for our networks:

|  | Ownership | Usage/ Interaction | Domain | Followers + Enemies | Parental | Co - Work | Conflict | Total |
|---|---|---|---|---|---|---|---|---|
| **Case 1: 100** | 14.7**KB** | 410.2**KB** | 44.7**KB** | 2.4**MB** | 8.9**KB** | 171.5**KB** | 84.3**KB** | 5.8**MB** |
| **Case 2: 500** | 82.4**KB** | 75.1**MB** | 1.1**MB** | 548**MB** | 45**KB** | 4.5**MB** | 2.2**MB** | 630.9**MB** |
| **Case 3: 1000** | 162.5**KB** | 579.1**MB** | 4.4**MB** | 4.4**GB** | 89.5**KB** | 18.2**MB** | 8.8**MB** | 5.0**GB** |

***Table*** *5.5.9: Disk Storage Requirements for GraphML files*

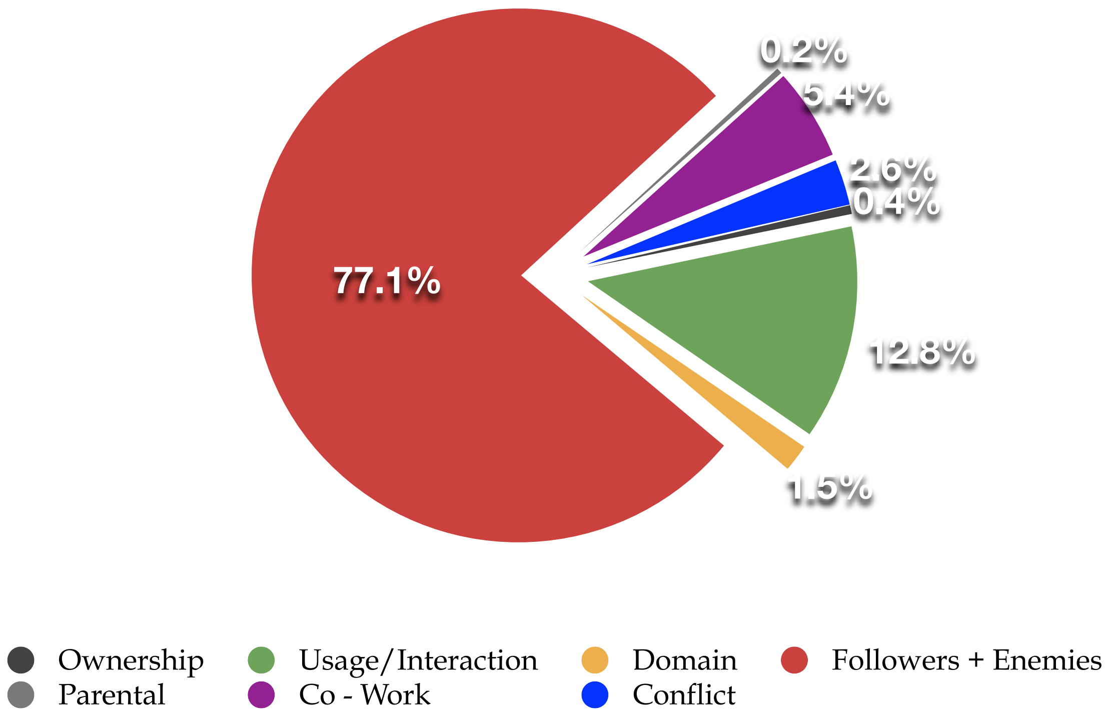


***Figure*** *5.5.9: Disk Storage Percentages for Case 1 files*

It is immediately evident that the most prominent relationships, in terms of space storage, are Followers/Followees and Enemies Object Relationships. Combined, they amount to 77.1% of the total size of complete network. Another 12.8% is allocated to Usage/Interaction. This is easily confirmed when we look back at the Case 1 statistics, where more than half of the total amount relationships are Followers/Followees and Enemies Relationships, followed by he Usage/Interaction relationships. These percentages are directly affected by two factors: a) The maximum number of followers and enemies a VE can have is set to 60% of the network size and b) The maximum number of users a VE can have is set to 60% of the network size. Lowering those thresholds can produce smaller files , as well as shorter running times.

# References

[1] *Relational Models Theory* [Online]. Available: http://www.rmt.ucla.edu.

[2] S. Higginbotham, "Amazon's Internet-of-things strategy takes shape," 31-Mar-2015. [Online]. Available: http://fortune.com/2015/03/31/amazons-internet-of-things-strategy-takes-shape/.

[3] R. Miller, "IBM Launches Major Internet of Things Offensive." [Online]. Available: http://social.techcrunch.com/2015/03/30/ibm-wants-to-get-head-start-on-internet-of-things/.

[4] "Moving from insight to action with Azure IoT services" [Online]. Available: http://www.microsoft.com/en-us/server-cloud/customer-stories/rockwell-automation.aspx.

[5] "Giving the world's cities a lift with IoT" *microsoft.com*. [Online]. Available: http://www.microsoft.com/en-us/server-cloud/customer-stories/Thyssen-Krupp-Elevator.aspx.

[6] "Autolib' drives the value of IoT" *microsoft.com*. [Online]. Available: http://www.microsoft.com/en-us/server-cloud/customer-stories/autolib.aspx.

[7] T. J. McCue, "$117 Billion Market For Internet of Things In Healthcare By 2020," *forbes.com*. [Online]. Available: http://www.forbes.com/sites/tjmccue/2015/04/22/117-billion-market-for-internet-of-things-in-healthcare-by-2020/.

[8] "The COSMOS project," *iot-cosmos.eu*. [Online]. Available: http://iot-cosmos.eu/.

[9] S. Wasserman, *Social Network Analysis*. Cambridge: Cambridge University Press, 1994.

[10] D. M. boyd and N. B. Ellison, "Social Network Sites: Definition, History, and Scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, Dec. 2007.

[11] C. Brown, N. Lathia, A. Noulas, C. Mascolo, and V. Blondel, "Group colocation behavior in technological social networks," *arXiv.org*, vol. cs.SI. 07-Aug-2014.

[12] M. Naaman, "Is it Really About Me? Message Content in SocialAwareness Streams," Sep. 2009.

[13] K. Lerman and R. Ghosh, "Information Contagion: an Empirical Study of the Spread of News on Digg and Twitter Social Networks," *arXiv*, vol. cs.CY, 2010.

[14] S. Dashti, L. Palen, M. P. Heris, K. M. Anderson, S. Anderson, and T. J. Anderson, "Supporting Disaster Reconnaissance with Social Media Data:A Design-Oriented Case Study of the 2013 Colorado Floods," Feb. 2014.

[15] J. B. Nash, *Spectatoritis*. 1932.

[16] G. G. Roberts, *Dick Tracy and American Culture*. McFarland, 1993.

[17] "Wonders of Modern Technology: Barcodes Sweep the World."

[18] M. L. HEILIG, "Stereoscopic-television apparatus for individual use," US2955156.

[19] "internet_first_words.html," *lk.cs.ucla.edu*. [Online]. Available: http://www.lk.cs.ucla.edu/internet_first_words.html.

[20] M. Cardullo, "Genesis of the Versatile RFID Tag," *rfidjournal.com*. [Online]. Available: http://www.rfidjournal.com/articles/pdf?392.

[21] "CMU SCS Coke Machine," *cs.cmu.edu*. [Online]. Available: http://www.cs.cmu.edu/~coke/.

[22] M. Weiser, "The Computer for the 21st Century," *wiki.daimi.au.dk*. [Online]. Available: http://wiki.daimi.au.dk/pca/_files/weiser-orig.pdf.

[23] M. Lamming and M. Flynn, "Forget-me-not: Intimate computing in support of human memory," *Proc FRIEND21*, 1994.

[24] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *Network, IEEE*, vol. 8, no. 5, pp. 22–32, 1994.
[25] S. Sarma, D. L. Brock, and K. Ashton, "The Networked Physical World."
[26] D. Giusto, A. Iera, G. Morabito, and L. Atzori, *The Internet of Things*. New York, NY: Springer Science & Business Media, 2010.
[27] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
[28] M. Presser and A. Gluhak, *The Internet of Things: Connecting the Real World with the Digital World*. EURESCOM mess@ ge–The Magazine for Telecom …, 2009.
[29] J. P. Vasseur and A. Dunkels, "IP for smart objects," *IPSO Alliance*, 2008.
[30] I. Toma, E. Simperl, and G. Hench, *A joint roadmap for semantic technologies and the internet of things*. Proceedings of the Third STI Roadmapping Workshop, 2009.
[31] H. Vogt, "Efficient Object Identification with Passive RFID Tags," in *Pervasive Computing*, vol. 2414, no. 9, Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 98–113.
[32] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.
[33] "Catalog of IoT Domains for Applications," pp. 1–41, Oct. 2014.
[34] "Nest Thermostat: What is Auto-Away?."
[35] "Google to Acquire Nest," *investor.google.com*.
[36] "Mimo Smart Baby Moitor," *mimobaby.com*. [Online]. Available: http://mimobaby.com/.
[37] "Bigbelly Waste & Recycling Stations," *bigbelly.com*. [Online]. Available: http://bigbelly.com/solutions/stations/.
[38] "Streetline: ParkSight 2.0 Parking Analytics," *streetline.com*. [Online]. Available: http://www.streetline.com/parking-analytics/.
[39] "Lighting City of Oslo: Street lighting case study," presented at the echelon.com.
[40] "Lighting: Street Lighting Paris Senart case study," presented at the echelon.com.
[41] L. Atzori, A. Iera, and G. Morabito, "SIoT: Giving a Social Structure to the Internet of Things," *Communications Letters, IEEE*, vol. 15, no. 11, pp. 1193–1195, Nov. 2011.
[42] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl5, and H.-W. Gellersen, "Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts," in *Ubicomp 2001: Ubiquitous Computing*, vol. 2201, no. 10, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 116–122.
[43] J. Bleecker, "A Manifesto for Networked Objects — Cohabiting with Pigeons, Arphids and Aibos in the Internet of Things," *(2006)*, 2006.
[44] E. Nazzi and T. Sokoler, *Walky for embodied microblogging: sharing mundane activities through augmented everyday objects*. New York, New York, USA: ACM, 2011, pp. 563–568.
[45] M. Kranz and L. Roalter, "Things that twitter: social networks and the internet of things," *… the Internet of Things do …*, 2010.
[46] P. Mendes, "Social-driven internet of connected objects," *Proc of the Interconn Smart Objects with the Internet …*, 2011.
[47] J. Porcello, "From "Smart Objects" to "Social Objects": The Next Evolutionary Step of the Internet of Things," pp. 1–9, Nov. 2014.
[48] A. P. Fiske, *Structures of social life: The four elementary forms of human relations: Communal sharing, authority ranking, equality matching, market pricing*. Free Press, 1991.
[49] "Internet Encyclopedia of Philosophy: Relational Models Theory," *iep.utm.edu*. [Online]. Available: http://www.iep.utm.edu/r-models/.
[50] A. Computing, "An architectural blueprint for autonomic computing," *IBM White Paper*, 2006.

[51] O. Voutyras, P. Bourelos, D. Kyriazis, and T. Varvarigou, "An Architecture supporting Knowledge flow in Social Internet of Things systems," Oct. 2014.
[52] O. Voutyras, S. V. Gogouvitis, A. Marinakis, and T. Varvarigou, "Achieving Autonomicity in IoT systems via Situational-Aware, Cognitive and Social Things," presented at the the 18th Panhellenic Conference, New York, New York, USA, 2014, pp. 1–2.
[53] J. E. Rowley, "The wisdom hierarchy: representations of the DIKW hierarchy," *Journal of Information Science*, vol. 33, no. 2, pp. 163–180, Feb. 2007.
[54] L. Atzori, A. Iera, and G. Morabito, "Making things socialize in the Internet — Does it help our lives?," presented at the Kaleidoscope 2011: The Fully Networked Human? - Innovations for Future Networks and Services (K-2011), Proceedings of ITU, 2011, pp. 1–8.
[55] O. Voutyras, P. Bourelos, S. Gogouvitis, D. Kyriazis, and T. Varvarigou, "Social Monitoring and Social Analysis in Internet of Things Virtual Networks," pp. 1–8, Jan. 2015.
[56] "Gephi Main Page," *gephi.github.io*. [Online]. Available: http://gephi.github.io/.
[57] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software" *plosone.org*. [Online]. Available: http://www.plosone.org/article/fetchObject.action?uri=info:doi/10.1371/journal.pone.0098679&representation=PDF.
[58] T. M. J. Fruchterman and E. M. Reingold, "Graph Drawing by Force-directed Placement," pp. 1–36, Jan. 1997.