CARDIFF UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & INFORMATICS

Policy-based Asset Sharing in Collaborative Environments

Author

Christos PARIZAS

Advisor

Prof. Alun D. PREECE

September 2015

A thesis submitted in partial fulfillment

of the requirement for the degree of Doctor of Philosophy

Abstract

Resource sharing is an important but complex problem to be solved. The problem is exacerbated in a dynamic coalition context, due to multi-partner constraints (imposed by security, privacy and general operational issues) placed on the resources. Take for example scenarios such as emergency response operations, corporate collaborative environments, or even short-lived opportunistic networks, where multi-party teams are formed, utilizing and sharing their own resources in order to support collective endeavors, which otherwise would be difficult, if not impossible, to achieve by a single party. Policy-Based Management Systems (PBMS) have been proposed as a suitable paradigm to reduce this complexity and provide a means for effective resource sharing.

The overarching problem that this thesis deals with, is the development of PBMS techniques and technologies that will allow in a dynamic and transparent way, users that operate in collaborative environments to share their assets through high-level policies. To do so, it focuses on three sub-problems each one of which is related to a different aspect of a PBMS, making three key contributions. The first is a novel model, which proposes an alternative way for asset sharing, better fit than the traditional approaches when dealing with collaborative and dynamic environments. In order for all of the existing asset sharing approaches to comply with situational changes, an extra overhead is needed due to the fact that the decision making centre – and therefore, the policy making centre – is far away from where the changes take place unlike the event-driven approach proposed in this thesis.

The second contribution is the proposal of an efficient, high-level policy conflict analysis mechanism, that provides a more transparent – in terms of user interaction – alternative way for maintaining unconflicted PBMS. Its discrete and sequential execution, breaks down the analysis process into discrete steps, making the conflict analysis more efficient compared to existing approaches, while eases human policy authors to track the whole process interfacing with it, in a near to natural language representation.

The contribution of the third piece of research work is an interest-based policy negotiation mechanism, for enhancing asset sharing while promoting collaboration in coalition environments. The enabling technology for achieving the last two contributions (contribution 2 & 3) is a controlled natural language representation, which is used for defining a policy language. For evaluating the proposed ideas, in the first and third contributions we run simulation experiments while we simulate and also conduct formal analysis for the second one.

Key words: Policy-based Asset Sharing, Dynamic Asset Sharing, Policy Language, Efficient Policy Conflict Analysis, Interest-based Policy Negotiation

Email: C.Parizas@cardiff.ac.uk

WWW: `https://users.cs.cf.ac.uk/C.Parizas/`

Acknowledgements

This research was sponsored by the US Army Research Laboratory and the UK Ministry of Defense and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defense or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes not with standing any copy right notation hereon.

I would like to express my sincere gratitude to my advisor, Professor Alun Preece who gave me the opportunity to pursue a PhD. You have been a tremendous mentor for me and helped me to grow up professionally. Your deep knowledge and assistance and this feeling that you have always been there as my closest teammate and supporter, have been invaluable for completing my thesis. Working at your side and learning from your wide research experience and ability to manage research problems, helped me to mature as a researcher. I knew it before starting my PhD that whoever my advisor would be, he or she would be influencing my current and future professional life. I am deeply thankful that this person was you Alun. A deep thanks also to all of your patience in working with me.

A special thank you to my three great mentors in this journey who gave me the opportunity to work as a research intern the summers of 2013 and 2014; Dr. Geeth de

Mel and Dr. Petros Zerfos of IBM Watson Research Center and Dr. Tien Pham in U.S. Army Research Laboratory. Both internships were huge experiences and milestones for my development as a researcher and helped me to understand how it is to work in high standards with one of the best industrial and governmental research labs in US. I was also very lucky with my internships to meet exceptional research students from top universities around the world who soon are going to be IT leaders in academia and industry. I am glad I can name many of them my friends.

Thanks to Dr. George Theodorakopoulos who has been for the past two years my “big research brother", who through our daily and insightful discussions around CS research topics inculcated me valuable research principles. A big thank you also to Dr. Diego Pizzocaro who has been my “junior advisor" during the first year of my PhD and the person who got me – hands on – into research.

I consider myself very lucky to be involved in a project such as, the International Technology Alliance. I had the privilege, besides the bright folks that I had close collaboration with, to meet world-class researchers from both academic and industrial research institutions. In full gratitude I would like to acknowledge the following individuals and my co-authors who encouraged, inspired, supported and assisted me throughout these years: Dr. Murat Sensoy (Ozyegin University), Dr. Yousaf Shah (IBM Watson), Dominic Harries (IBM UK Hursley), Dave Braines (IBM UK Hursley), Dr. John Ibbotson (IBM UK Hursley), Prof. Boleslaw Szymanski (RPI), Dr. Christopher Gibson (IBM UK Hursley) and Dr. Seraphin Calo (IBM Watson).

I would like to thank Dr. Gregorios Loukides and Dr. Christine Mumford who as my inner school, yearly review panelists, provided me and supported my research with helpful comments and feedback. I owe a thank you also to all my lab mates throughout these years, who made the boring and tough days in the office a bit more affordable.

A big thank you to Foroula Grivaki, who was my first supporter in leaving my country and pursuing a postgraduate degree abroad.

Finally, the greatest thanks are for my beloved family; my mother Katerina, my father Vagelis and my brother Stergios for their unconditional support and encouragement. They have always been there for me, on the best and worst times.

Thank you all folks!

_Christos

Contents

List of Publications

The work introduced in this thesis is based on the following publications.

Chapter 2:

- C. Parizas, D. Pizzocaro, A. Preece, P. Zerfos. Sharing Policies for Multi-Partner Asset Management in Smart Environments. In proceedings of the IEEE Network Operations and Management Symposium, NOMS 2014
- C. Parizas, D. Pizzocaro, A. Preece, P. Zerfos. Sharing Smart Environment Assets in Dynamic Multi-Partner Scenarios. In proceedings of the Sixth IEEE/IFIP International Workshop on Management of the Future Internet, ManFI, 2014

Chapter 3:

- C. Parizas, D. Pizzocaro, A. Preece, P. Zerfos. Managing ISR sharing policies at the network edge using Controlled English. In proceedings of the Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IV, Baltimore, USA, 2013
- C. Parizas, A. Preece, P. Zerfos, S.Y. Shah, B. Szymanski, C. Gibson, D. Harries. Leveraging Controlled English at the Network Edge for Policy-Based Management of ISR Services. In proceedings of the 1st Annual Fall Meeting of ITA, Palisades, NY, USA, 2013

- S.Y. Shah, C. Parizas, Geeth De Mel, B. Szymanski, A. Preece, D. Harries, J. Ibbotson, S. Pipes. Multi-layer Human-in-the-loop Service Management Utilizing Rule-based Policies and Set Theoretic Expressions. In proceedings of the 2nd Annual Fall Meeting of ITA, Cardiff, UK, 2014

Chapter 4:

- C. Parizas and P. Zerfos. CE-based policies conflict analysis for sensor services management in information fabric. Palisades, NY USA, 2013. In proceedings of the 1st Annual Fall Meeting of ITA.
- C. Parizas, P. Zerfos, A. Preece, Geeth De Mel. High-level Policy Conflict Analysis at the Network Edge – To be published

Chapter 5:

- C. Parizas, G. de Mel, A. Preece, M. Sensoy, S. Calo and T. Pham. Interest-based Negotiation for Asset Sharing Policies, in Proceedings of the Coordination. In proceedings of the Organizations, Institutions and Norms in Agent Systems (COIN 2015), 12th International Conference on Autonomous Agents and Multiagent Systems, 2015.

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ABN Argumentation-based negotiation

AO Area of Operation

C2 Command and Control

CE Controlled English

CIM-SPL Common Information Model Simple Policy Language

CNL Controlled Natural Language

CoI Communities of Interest

COPS Common Open Policy Service

DMTF Distributed Management Task Force

ECA Event, Condition, Action

FOF First-order Logic

GUI Graphical User Interface

IBN Interest-based negotiation

ISR Intelligence, Surveillance and Reconnaissance

IETF Internet Engineering Task Force

ITA	International Technology Alliance
KAOS	Knowledge Acquisition in automated Specification
LDAP	Lightweight Directory Access Protocol
MAS	Multi-agent System
MSTA	Multi-Sensor Task Allocation
NL	Natural Language
OWL	Web Ontology Language
P2P	Peer-to-Peer
PBMS	Policy-based Management Systems
PBN	Position-based Negotiations
PDL	Policy Description Language
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PNP	Policy Negotiation Point
PR	Policy Repository
QoS	Quality of Service
SAT	Subject, Action, Target
SMD	Smartphone Device
SNMP	Simple Network Management Protocol
TR	Team Range

WPML	Watson Policy Management Library
XACML	eXtensible Access Control Markup Language

Chapter 1

Introduction

The monitoring, control and configuration of large-scale distributed systems and provided services is a complex and daunting affair. Policy-based management systems (PBMS) have been proposed as a suitable paradigm to reduce this complexity. PBMS have been broadly researched the last 20 years exploring their applications in a variety of IT management tasks such as resource access control, database configuration, network administration and provided services planning [15, 59, 97, 107].

Policy rules in the majority of PBMS applications reflect management restrictions with respect to domains such as security, privacy and performance to name only a few. Their main emphasis, as probably could be expected, is on security aspects. A PBMS is rather an abstract term. A general definition in the context of our research domain would be that of *a systems' management framework that consists of two major parts: a) a set of behavioral and functional rules that the managed system must comply with and b) a set of this very system's resources capable of interpreting and enforcing those rules to themselves.*

From an operational point of view, a PBMS provides the IT administrators with an abstraction layer that describes the system to be managed, enabling them to express their high level management goals and objectives through high level policy rules, instead of trying to manage the system at the individual resource level, through fine-grained, targeted scripts. Moreover, it provides a more flexible paradigm for systems management allowing administrators to interfere with them by changing the policies and not their underlying implementation. Finally, the ultimate goal of a PBMS is to provide a technological layer for making systems automated administration a reality.

1.1 Policy Rules

The cornerstone of any PBMS is the policy rules. Many definitions have been proposed throughout the years trying to capture what a *policy* is (in the context of PBMS, not from sociological, political and other viewpoints that might have different and broader definitions). Our definition of policy in the narrowed spectrum of PBMS is this of *a guide for a system's actions, towards behaviors that would secure optimal system outcomes.* This definition makes clear the separation between management layers that can be expressed through policies referring to both possible layers; a) the more technical extension of policy term which is expressed through the *"actions"* term of definition, and b) the *"behaviors"* term that describes the higher/business level management that can be described and enforced through policies. This separation is not binary as policies can be represented at different levels of abstraction (from very technical, to very business oriented). Hence, a PBMS can be composed of policies that range from abstract business goals to device-specific configuration parameters definition.

The four basic grammatical blocks of a policy rule, as described in the majority of the literature and standardization organizations are the policy's *Subject, Target, Action and Condition(s)* [64].

- **Subject** is the entity, or a collection of them, that originates an Action.
- **Target** is the entity, or collection of them, that is affected by a policy.
- **Action** entity refers to what is to be done on a system, if the conditions of the policy rule are met.
- **Condition(s)** describe the necessary system state and/or general prerequisites that define whether a policy rule's actions should be performed.

The PBMS' policies are the binding of a set of *Actions* to a set of *Conditions*, where the *Conditions* are evaluated to determine whether policy *Actions* should be performed on policies' *Targets* by policies' *Subjects*.

Policies are usually expressed in formal, textual languages called *policy languages*. There is a plethora of policy languages proposed in the literature such as, [24, 26, 55, 69], with different characteristics and formalities based on the purpose that the PBMS they are integrated in intends to manage. A major classification of proposed policy languages lies on whether they support special or more general purpose PBMS.

1.2 PBMS Architecture

Apart from the language representation, two other basic building blocks are necessary for the implementation of a PBMS. The first is a formalism to describe the managed system and the second is the reasoning mechanism in charge of evaluating policies on this representation. The formal description of the managed system is the abstract representation of the entities in it and their properties. It also describes system operations formed as sets of relationships between the system's entities. They are usually expressed through information models or utilizing formal logic representations.

We close the general introduction on PBMS by presenting its architecture as it has been proposed by standardization authorities such as DMTF and IETF and presented in Figure 1.1 [106]. A PBMS, consists of four basic logical components:

1. The policy management tool,
2. The policy repository (PR),
3. The policy enforcement point (PEP), and
4. The policy decision point (PDP)

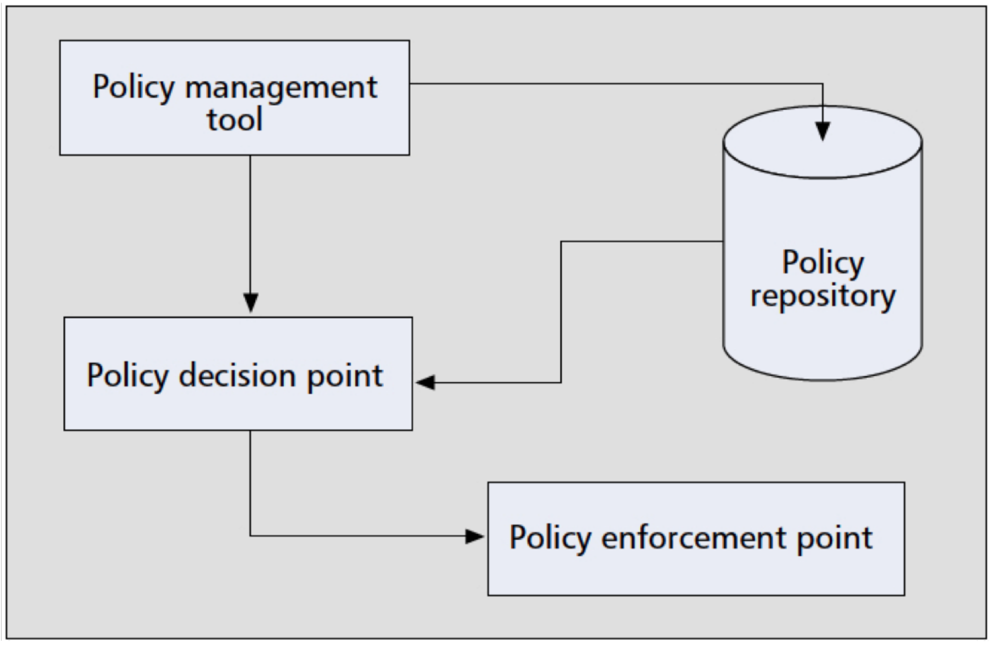


Figure 1.1: PBMS Architecture

The policy management tool is the entry point through which policy makers interface (i.e., write, update and delete) with policies to be enforced on the system. The policy repository is the data storage element where the policies generated by the management tool are held. The PEP is the logical component that can take actions on enforcing the policies' decisions, while the PDP is the entity that makes these decisions either for itself or for other systems components if it is requested to. As for the PBMS' operational flow, once policy rules are developed by policy authors they are stored in the PR. Triggered by an event that needs for specific policy rules to be evaluated, the PEP contacts PDP which is responsible for fetching the necessary ones from the PR. Then it evaluates them and decides the actions that need to be enforced on PEP.

Common Open Policy Service (COPS) [33] and Simple Network Management Protocol (SNMP) [18] are two of the most well-known protocols for establishing communication between PBMS' logical components. The Lightweight Directory Access Protocol (LDAP) [108] has been also proposed and widely used as a suitable protocol for allowing PDP accessing and general maintaining distributed directory information in PR.

Although the development of a policy-based management framework is outside the scope of this thesis, it is worth to mention here one of the most well-known policy-based management framework, developed by researchers in Imperial College. The work in [25, 32] describes the development of the Ponder policy-based toolkit which provides a platform, based on object-oriented Java implementation. The view of Ponder regarding the concept of a policy is that of an object that describes relationships between subject and target policy blocks. Ponder is maybe the most widely discussed toolkit in the policy-based research community and providing several components it support the entire life-cycle of a policy, covering its development, specification and deployment. Other well-known policy frameworks are the PMAC platform which proposes a generic middleware framework for managing aspects of large-scale distributed systems [4] and the Watson Policy Management Library (WPML), which provides a model and APIs to enable the easy development of software platforms that use policies to define access rights, filtering and any other operation that needs to be dynamically deployed into the enabled platform [102]. Both frameworks were deployed in IBM Watson Research Lab.

1.3 Thesis Overarching Problem

The overarching problem this thesis deals with is the development of PBMS technologies that will allow in a transparent and effective way users that operate in collaborative environments to share their assets through high-level policies. To do so, it focuses on four sub-problems; each one of which deals with a different aspect of PBMS' operation. Two of the sub-problems (sub-problems 2 & 3) constitute major challenges for the policy-based management community over the last two decades, while the other two (sub-problems 1 & 3) got relatively less attention. However, due to the importance in achieving our primary goal, it was imperative to us to research them. The four sub-problems are as follows:

1. A Team-centric policy-based asset sharing in dynamic and collaborative environments as presented in Chapter 2.
2. A user-friendly policy language as presented in Chapter 3.
3. An efficient policy conflict analysis mechanism as presented in Chapter 4.
4. An interest-based policy negotiation mechanism for enhancing asset sharing, promoting collaboration as presented in Chapter 5.

The key contributions of this thesis are as follows:

1. The proposal of a novel, team-centric model suitable for high-level asset sharing in dynamic collaborative environments.
2. The proposal of an efficient mechanism for dealing with the complex task of policy conflict analysis, in dynamic collaborative environments.
3. The proposal of a novel interest-based policy negotiation mechanism, for enhancing asset sharing while promoting collaboration in dynamic collaborative environments.

We do not consider the proposal of the user-friendly policy language as one of our key contributions; it mostly provides the enabling technology to our main contributions. For the evaluation of the proposed mechanisms, in the cases of contributions 1 and 3 we run simulation experiments while for contribution 2 we conduct formal analysis of the proposed model's algorithms as well. The following section describes in more details the research motivation behind the four pieces of research work and their contribution.

1.4 Research Motivation & Contribution

This thesis deals with four aspects of policy-based management, of systems that operate in dynamic and collaborative environments. This section frames the domain of our research interest, the motivation behind all four pieces of work and it also highlights research contribution. The Introduction Chapter finally, closes presenting in Section 1.5 the structure of the rest of the thesis.

1.4.1 PBMS in Dynamic and Collaborative Environments

What a collaborative and dynamic environment is and where do we meet those environments? Think of unexpected scenarios such as emergency response and humanitarian relief operations, where two or more organizations merge forces and form coalitions to achieve common scenario objectives. Or military operations where multi-national and organizational groups collaborate in order for an operation to be successful. Even the corporate environments, where businesses collaborate sharing for instance, IT services and other resources aiming to achieve higher profits or lower operational costs. Or finally, short-lived, mobile, opportunistic networks comprised of peer members, established for message routing or data sharing. All the above constitute dynamic and complex environments in terms of their management and hence, environments that a policy-based management paradigm would be a good fit for. A dynamic environment is characterized by constant change, activity or progress. Moreover, each one of them rely more or less on multi-party collaboration. The policy-based management, of systems that operate in scenarios such as the aforementioned are the source of our research motivation.

What kind of systems are utilized by collaborating parties on those scenarios and hence on what kind of systems does our policy-based management technologies apply? In emergency response situations supportive systems might include smart environment ecosystems composed of different IT assets such as sensing devices and data storage. In military scenarios the managed systems might be any kind of Intelligence, Surveillance and Reconnaissance (ISR) infrastructures, while in corporate scenarios, this can be a variety of hybrid cloud services. As it might be inferred the word “system" does not set any limitation on whether it refers to software or hardware infrastructures. Those generic systems can include hard/IT resources as well as soft/human resources too.

The following three Subsections present the research motivation and contribution of all three research pieces we deal with, while the fourth one presents the enabling technology for achieving contributions 2 and 3. Each one of the Subsections utilizes scenarios such as those described above. We do so to show the broad variety of applications our research has in covering different PBMS aspects on a variety of policy-based collaborative environments' management.

1.4.2 Team-centric Asset Sharing in Collaborative Environments

Motivation

Advances in information technology including the Internet, sensors and communication protocols, combined with the availability of cheaper computing power set the future for intelligent, interconnected and interacting “smart environments”. Smart environments are ecosystems composed of infrastructures that blend physical and IT assets, wherein sensors, network connectivity and data storage are embedded seamlessly in physical objects such as power grids and buildings, and hold promise for improving the efficiency of our social and economic ecosystems [100]. Smart environments come as a result of an effective integration of planning, construction and management methods prepared for both expected and unexpected events [91].

In unexpected scenarios such as humanitarian relief and emergency response operations, two or more organizations that own and operate disparate sets of IT and infrastructure assets merge forces and form coalitions to achieve common scenario objectives. In such cases, facilitating the collaboration between multiple partners and establishing trust, while considering issues such as security and privacy, is a precondition for the successful formation and operation of the multi-organization partnership. Typically, collaborating partners have their own inherent restrictions, which are stated as a set of policies including for instance security, privacy and legal policies on how to share their infrastructures with other organizations.

Recent examples of natural disaster situations, such as those that unfolded during this year's Nepal earthquake[1], the 2011 Fukushima Daiichi nuclear disaster[2], and the damaged utility facilities in the BP oil spill case[3] demonstrated the need for emergency responders such as first aiders, rescuers and engineers, affiliated with different national and organizational groups to form cross-organizational teams and share assets in an ad-hoc manner, in order to provide humanitarian assistance.

Within such stressed environments, assets such as sensors, network and service monitoring systems, as well as data analytics and inference engines, collect, process and disseminate operational data and insights. These assets are shared across organizations to enable quick decision making.

Sharing smart environment assets to support multiple concurrent and multi-organization missions is a non-trivial problem. Collaborating organizations have different backgrounds (e.g. area of expertise, cultural background) which reduce shared awareness and understanding of the mission, leading to different decisions about what assets can be shared, with whom, and when [2]. Moreover, asset sharing is a time critical process, given the highly dynamic environments, thus it needs to be characterized by agility.

Contribution

One of the most widespread asset sharing models is the static asset ownership approach, according to which assets belonging to a collaborative partner may or may not be shared with other partners, based on pre-defined policies [38]. We refer to this

[1]Nepal earthquake - `http://tinyurl.com/nufg8zg`
[2]Fukushima Daiichi nuclear disaster - `http://tinyurl.com/pmqx3lr`
[3]Deepwater Horizon oil spill - `http://tinyurl.com/d6qjkfa`

as *asset-centric* sharing model. However, this approach usually assumes a centralized operation and control center; the decision makers/policy makers in this centralized approach, are senior personnel located away from where the situational changes take place. In order to consider for instance a policy update, given the changes occurring on the operational field, they need to get some input/feedback through vertical communication from lower rank users operating at the edge of the network. Thus, the asset-centric sharing model limits the formation of ad hoc decision making, regarding asset sharing in which membership of assets is highly dynamic and changes frequently. This limitation is further exacerbated in the case where communication to the centralized operations center is intermittent and unreliable.

The first piece of research work in this thesis, proposes a novel asset sharing scheme based on the *Edge C2* model [43]. According to this scheme, members from multiple organizations are grouped into cross-organizational teams and share ownership, monitoring and management authority of these assets among themselves, regardless of their organizational affiliation. We call this the *team-centric* sharing model. This approach is made feasible nowadays due to the availability of ad hoc network connectivity [40] and distributed middleware infrastructure [104], which allow for direct communication and management of assets and services that belong to disparate administrative domains. This new asset sharing approach shifts decision making power regarding access and control of assets to the edge of organizations [53], thus allowing for more dynamic formation of teams and assets sharing patterns to emerge.

In [77], Pizzocaro et al. proposed a distributed protocol, for addressing the problem of allocating a heterogeneous bundle of sensing assets to a variety of different sensing tasks, with the goal of maximizing the usefulness of assets and satisfying the most critical task requests. They refer to this problem as Multi-Sensor Task Allocation (MSTA). While MSTA was initially designed for sensors, by considering the sensors as sensing services and the sensor bundles as composite services, the algorithm is generally applicable to services-to-task allocation. We formalize, evaluate and compare the two aforementioned asset sharing models, investigating their impact on MSTA-P; an asset sharing, policy-enabled version of MSTA protocol. MSTA-P is used herein, as a metric for measuring the performance of asset-centric and team-centric sharing models and is presented in detail in the following section. In particular, this first piece of work makes the following contributions:

1. A formal representation of the two asset sharing models, (asset-centric & team-centric) using predicate logic, which makes them amenable to analysis.
2. The policy-regulated version of asset-task assignment protocol called MSTA-P, which integrates asset sharing policies evaluation.
3. Evaluation and comparison of the asset-centric and novel team-centric asset sharing models using a discrete-time, multi-agent, simulation environment.

We find that while the traditional ownership model allows slightly better performance, the difference is only marginal, so a team-sharing model offers a viable alternative sharing approach.

1.4.3 Controlled Natural Language Policy Representation

In this subsection we make an introduction to the enabling technology for achieving contributions 2 and 3 as described in Chapters 4 and 5 respectively, presenting it through the spectrum of a military scenario.

In military operations accurate, reliable and actionable intelligence is needed in order for those operations to be successful. This intelligence is increasingly produced by ISR systems, which provide key capabilities to the command authorities for intelligence collection, exploitation and battle management. An ISR system can be for example used for border reconnaissance and surveillance or object detection and localization. In aforementioned environments, ad-hoc Communities of Interest (CoIs) act together to achieve a set of common objectives forming coalitions. Recent history shows that coalition operations are going to be the norm, not the exception [60]. A coalition is a set of organizations that work together usually in peer-to-peer formations where through collective actions, they are able to jointly perform tasks that they would not be able to perform or perform poorly otherwise [49]. Several issues emerge due to the multi-partner dimension of coalitions. Especially in military operations, where ISR resources are nationally-owned and operated systems, their sharing is often limited by security constraints [60], which is a defining factor in the collaboration between coalition partners.

An additional characteristic of the aforementioned environments is that they are highly dynamic. The decision makers operating in such an endeavor are usually required to deal with military, social, political and economic effects to name only a few, taking place simultaneously at the area of operation (AO) [8]. These effects are interrelated and make the environment that the decision makers want to influence more complex and dynamic and at the same time less understood and predictable. The need for agile management of ISR systems is pointed in several studies such as [96]. Several management constraints imposed by security, privacy and general operational issues of such resources in a coalition context are usually regulated by high-level policies enforced through PBMS.

A key step towards the development of PBMS that are able to quickly, easily and distributedly form, reform and negotiate resources management according to high frequency operational changes is to push policy development, and thus, the decision making center close to the source of situational changes in order to reduce the reaction time. The users who first cope with unexpected operational changes are those at or near the edge of the network, so we believe it would be beneficial if they were able to write, update and delete policy rules themselves without waiting for a central authority to approve each management request.

The network Edge C2 approach to designing command and control (C2) concepts, has

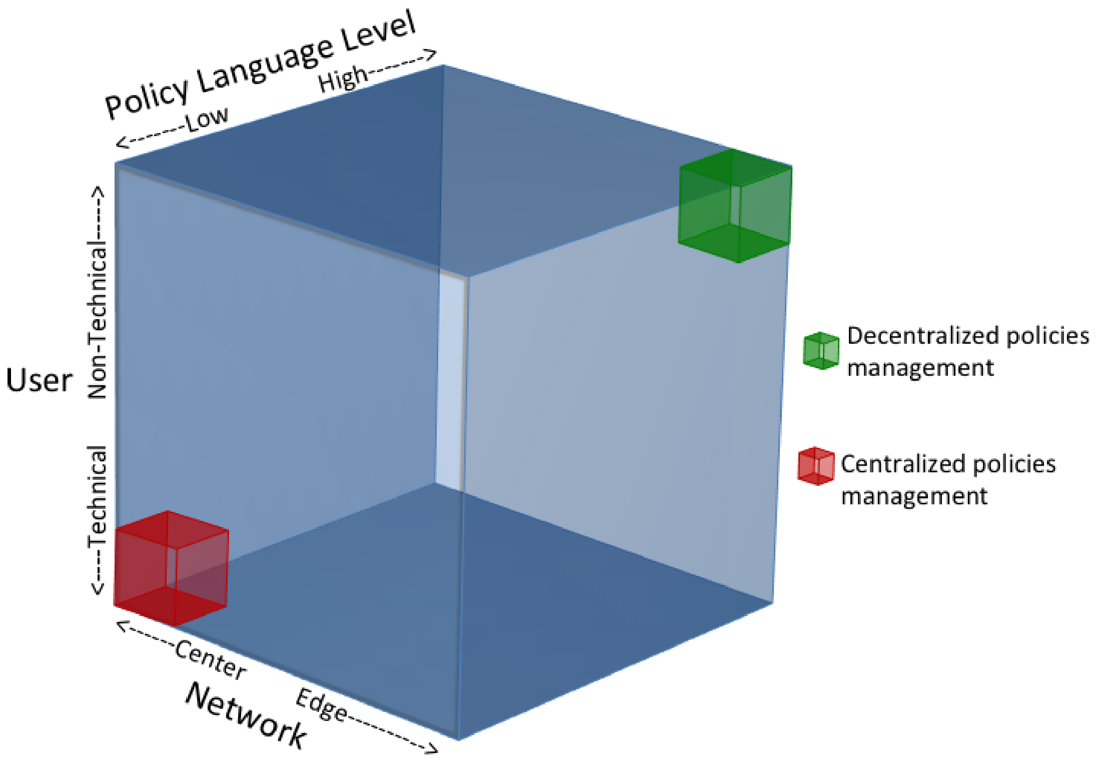


Figure 1.2: Pushing the policy-based management at the edge.

become popular in recent years as it involves the empowerment of individuals at the edge of the organization[43]. An emerging issue related to pushing decision making through policy formation at or near the edge, is the technical gap between existing low-level policy languages and non-technical users that operate in these areas. The vast majority of personnel at the organizational edge are not technology experts and so lack technical skills including the technical ability to maintain a PBMS through low-level policy languages. Well-known policy-based management systems such as the WPML[4] currently utilize low-level policy representation such as the Common Information Model Simple Policy Language[5] (CIM-SPL), making it difficult for non-Technical policy makers at the edge to easily maintain a PBMS.

[4]`http://tinyurl.com/od9zqno`
[5]`http://tinyurl.com/psetzwn`

In Figure 1.2 we portray the aforementioned hypothesis and present a centralized and the desired decentralized policy-based management approaches in the operation/organization network. The three axes of the big cube represent the three main features related to policy development in dynamic, coalition environments. First is the user types that operate in such environments in terms of their IT expertise, which varies from Technical to non-technical; second is the place in the organizational network where users operate, which can be anywhere from the center (e.g. a military base) to the very edge (e.g. warfare theater) and finally, the level of the policy language which might have either lower (i.e., technical language representation, easy to be used by IT experts) or higher (i.e., language representation close to natural language, easy to be understood by non-IT experts at the edge) user-friendliness characteristics.

The small cube at the left-bottom corner, represents the current state of PBMS, which are used by technical users, who operate near or at the center of the organizations, using low-level policy languages and are cumbersome for highly dynamic military environment. The small cube at the right-top corner, represents the problem space in which our contribution is situated; a PBMS, which is able to cope with highly dynamic environments by enabling the policies' development by non-technical users, who operate near or at the network edge, using user-friendly policy languages. The left-bottom corner cube represents a centralized directive management system based on the industrial age model, while the the right-top corner one represents a decentralized, emergent management system based on the information age model [88].

It is worth noting that the axes of the big cube in Figure 1.2 are not binary. The users can span from IT experts to users who lack any technical skills including those with different technical knowledge levels. We claim that a user-friendly policy can empower non-technical users (e.g. military planners and intelligence analysts), while at the same time cause no loss of technical users' expressiveness power (i.e., power provided via usage of low-level policy languages). As far as the organization network is concerned we focus on easing users with the maintenance of PBMS that operate at any place in between the military headquarters and the head of a battlefield operation.

We utilize CE as a means to define a policy representation that is both human-friendly and unambiguous for computers. CE is a human understandable version of Common Logic that is used here for expressing and enforcing a set of high-level (in terms of interface level with the user) management goals. CE and CE policy language are ontological-based models, that utilize semantics for representing domains and policy rules that govern them. The claim in [94], that a semantically oriented PBMS:

- eases the reduction of human error,
- simplifies policy analysis,
- reduces policy conflicts, and
- facilitates interoperability

reflects our point of view regarding formal representation of PBMS.

This piece of research work is to investigate whether CE is expressive enough to capture a variety of high-level, attribute-based, authorization policies related to a collaborative operation scenario and provides the enabling technology to our contributions regarding policy analysis and policy negotiation.

1.4.4 Policy Conflict Analysis in Collaborative Environments

Motivation

Despite the efforts of the last two decades from both academic and industrial sides, PBMS have not yet reached an adequate level of maturity that would make them applicable out of the box in scenarios such as the aforementioned ones. Thus, it is still difficult for system administrators to broadly taste the potential benefits that PBMS can bring towards effective and automated systems management. One of the reasons of PBMS' poor adoption is that it is difficult to develop and maintain them while guaranteeing integrity in terms of policy conflicts. Policy conflicts in general occur when two or more policies are applied simultaneously on the same resource and their resulting actions contradict with each other, leading to unpredictable system behavior. We can parallel them to the bugs in software development which are responsible for making systems to behave in unintended ways [58].

This thesis focuses on policy-based management of systems that operate in collaborative environments. The probability of conflicting policies' occurrence in such environments is high. The definition of a policy, as presented earlier in this chapter, is this of a guide for a system's actions, towards behaviors that would secure optimal system outcomes. In a collaborative environment the clear definition of an optimal system outcome, can be a challenging process, leading to conflicting policies. This may be, because partners come into collaboration with different perspectives, adding new dimensions to the process of collaboration; or due to the fact that different partners have their unique objectives and goals, which often try to keep them secret by other collaborators. Partners are also possible to have different backgrounds and expertise, or cultures and maybe speak different languages, which makes them sharing different understandings regarding systems' management. It is intuitively understood that multi-partner environments, implies a negative impact on the development and maintenance of unconflicted PBMS [7] especially when partners can be periodically involved in an operation, while it is possible for the same partners to be adversaries in another.

An additional characteristic of the environments that we look into is that they are highly dynamic. Therefore a dynamic policy-based management schema is imperative requiring dynamic policy rules writing and update. The complexity of highly dynamic environments cannot secure policy rules to be stable throughout the operations and requires them to change accordingly. As it is stated in [95] and it is also used as a conventional wisdom phrase *"with change comes conflict"*. Thus, the dynamic policy-based management schema adds an extra barrier on the implementation and maintenance of unconflicted PBMS making it a more challenging process, increasing the frequency of conflicting relationship between policies.

Finally, the users that operate on the environments that this thesis deals with, namely the policy authors are often not IT experts. Trying to reduce the response time in having valid and timely policy making, we encourage the push of the system's policy-based management near or at the network's edge where non-technical users operate. Consequently, an additional goal is the implementation of a policy conflict analysis mechanism that provides the means to non-IT experts to cope in a user friendly way with the complex task of conflict free PBMS maintenance.

Contribution

Responding to the characteristics of the environments that we look into, the second piece of research work presents a policy conflict analysis model, utilizing Controlled English (CE) [66]. CE is a form of Controlled Natural Language (CNL), namely a close to natural language representation, which is understood by English speaker users and processable by machines. In CE, domain model definition and policy rules can be precisely expressed while the CE reasoner enables the policy rules evaluation on the domain model. We utilize this CNL representation to propose a policy language as it is presented later.

As stated in [101], from a human input standpoint, a high-level policy language should be expressed in terms of natural language input. Adopting this opinion, a semiautomatic (i.e., a mechanism that needs input from human for its execution), policy conflict analysis tool, which interacts with human users through CNL representation, is a better fit especially when users are not technology experts.

Coping with highly dynamic and heterogeneous environments, in terms of policy authoring authorities, which consequently implies a potentially highly conflicting PBMS, we need a time efficient policy conflict analysis mechanism. Exploiting the semantics of CE and combining it with its hybrid ways of reasoning, provides us with a set of tools for developing time efficient, discrete and sequentially executed policy conflict analysis algorithms as presented in Chapter 4. Finally, the proposed policy conflict analysis tool is by nature discipline independent and can be used in generic PBMS middleware.

Summarizing the contribution of this second piece of research work, we propose an efficient and generic policy conflict analysis tool, capable of scaling better when the complexity of the managed system increases as far as the number of its entities is concerned compared to the naive pairwise approaches. Enumerating the pieces of contribution those are:

1. Proposal and theoretical analysis of CE-based policy conflict analysis algorithms.
2. Comparison of CE-based with existing approaches, focusing on execution time complexity.
3. CE-based policy conflict analysis prototype.

1.4.5 Interest-based Negotiation for Policy-based Asset Sharing

Motivation

Negotiation is a form of interaction usually expressed as a dialogue between two or more parties with conflicting interests that try to achieve mutual agreement about the exchange of scarce resources, resolve points of difference and craft outcomes that satisfy various interests. In order to cooperate and search for mutual agreements, the involved parties make proposals, trade options and offer concessions. The automation of the negotiation process and its integration with autonomic, multi-agent environments has been well-researched over the last few decades [46, 99].

The theoretical approaches for automated negotiation can be classified into three major categories: (1) game theoretic (2) heuristic, and (3) argumentation based [46]. The first two represent traditional, bilateral negotiation mechanisms wherein each negotiation party exchanges offers aiming to usually satisfy their own interests. Both approaches fall under the broader spectrum of position-based negotiations (PBN), where participants attack the opposing parties' offers, trying to convince them for the suitability of their own ones. Typically these approaches are formalized as search problems in the space of possible deals by focusing on negotiation objectives.

Argumentation-based negotiation (ABN) has been introduced as a means to enhance automated negotiation by exchanging richer information between negotiators. Interest-based negotiation (IBN) is a type of ABN that describes a mechanism where negotiating agents exchange information about the goals that motivate the negotiation action [37, 72]. IBN unlike PBN tackles the problem of negotiation by focusing on "why negotiate for" rather than on "what to negotiate for", aiming to lead negotiating parties to win-win solutions.

Multi-party teams are often formed to support collective endeavors, which otherwise would be difficult, if not impossible, to achieve by a single party. In order to support such activities, resources belonging to collaborating partners are shared among the team members. Mechanisms to share resources in this context are actively and broadly explored in the research community. This is due to the impact that different sharing modifications (what to share, with who, when and under what conditions) can bring into the collaboration, with respect to domains such as security, privacy and performance to name only a few.

Consider for instance the following scenarios: a) the resource sharing in corporate environments such as the recent *MobileFirst*[6] partnership between IBM and Apple where cloud and other services are shared in a daily basis; or b) a short-lived, mobile, opportunistic network comprised of a few peer members, established for message routing or data sharing. In all cases, access control mechanisms that specify resource sharing, need to be implemented for establishing smooth collaboration. A suitable mechanism

[6]http://www.ibm.com/mobilefirst/us/en/

for managing access control on resources of such systems is a policy-based management system (PBMS). The PBMS in general, provides systems administrators with a programable, abstraction layer of the system to be managed, enabling them to express high level management goals and objectives through high level policy rules. The more strict the partners' policy rules are, the higher the barriers towards collaboration are set. Thus, the need for a framework for enabling authorization policy negotiation in multi-party, cooperative and dynamic environments is imperative in order for those strict policy rules to be refined accordingly promoting collaboration.

Contribution

The third piece of research work presents a novel, interest-based policy negotiation mechanism for enabling authorization policy negotiation in multi-party, cooperative and dynamic environments. It focuses on policy makers who are not necessarily experts in either IT or negotiation techniques. To the best of our knowledge there is no mature work done on policy negotiation. The vast majority of automated negotiation work: a) deals with autonomous, multi-agent environments, b) utilizes PBN approaches and c) invariably ignores the special characteristics of multi-party, collaborative environments.

It is our belief that by understanding the interests behind collaborating parties' policies and by crafting options that can meet their asset sharing requirements, IBN could provide a negotiation mechanism, that promotes good collaboration unlike PBN, which inadvertently creates adversarial negotiation atmosphere. Moreover, the PBN paradigm with its fixed, opposing positions is a cumbersome negotiation method to cope with dynamic environments [46]. From an architectural point of view the proposed negotiation mechanism can operate in parallel to a PBMS. Briefly, the policy IBN considers an approach that proposes modification of strict policies, in order to increaseh overall usability of collaborators' assets while remaining faithful to existing authorization policies. The main contributions of this of work are as follows:

1. Definition of an interest-based authorization policy negotiation model.

2. Specification of an architecture for its integration with PBMS.
3. Evaluation of policy IBN behavior through simulation experiments.

1.5 Thesis Structure

Chapter 2 deals with the high-level, policy-based asset sharing in collaborative environments. In Section 2.2 we discuss previous work on asset sharing policies in multi-partner, collaborative operations and analyze our approach to emerged issues in the context of mobile, multi-agent environments. In Section 2.3 we formalize the proposed asset sharing policies, while in Section 2.4 we briefly present the MSTA, and we propose the policy-regulated MSTA-P protocol. Section 2.5 describes the experiments through which we evaluate and compare the sharing models using a simulation environment and in Section 2.6 we present the simulation's results. In Section 2.7 we discuss and conclude the chapter.

Chapter 3 presents the CE policy language. In Section 3.2 we present well known policy languages. In Section 3.4 we discuss the strengths of Edge C2 approach, we compare a CE-based policy language, in terms of user friendliness, with a well-known predecessor and we highlight the benefits of Edge C2 using CE as policy representation. In Section 3.5 we define and develop in CE a domain model to capture the multi-partner sharing aspects of a coalition operation. In Section 3.6 we demonstrate the expressiveness of CE as a policy language by developing and executing on the model a variety of attribute-based authorization rules.

Chapter 4 deals with CE-based policy conflict analysis. Section 4.2 presents prior work on policy conflict analysis and compares it to the proposed high-level approach. Section 4.3 presents the characteristics of high-level policy conflict analysis, it summarizes the CE policy language, classifies the potential policy conflicts and highlights the strength and weaknesses of the conflict analysis model. Section 4.4 demonstrates the conflict analysis' architecture, presents a theoretical analysis of prerequisites for policy conflicts occurrence and closes with the algorithmic steps. In Section 4.5 we describe the experiments through which we evaluate our model and present the evaluation results comparing it with a naive conflict analysis approach.

Chapter 5 copes with interest-based negotiation for policy-based asset sharing. Previous literature on policy negotiation approaches is discussed in Section 5.2, while Section 5.3 presents a walkthrough of a policy negotiation scenario. Section 5.4 describes the policy negotiation framework, the policy language, and its interface to PBMS by means of an architectural overview. Section 5.5 presents the algorithmic steps for IBN achievement through policy refinement and in Section 5.6 we evaluate the proposed IBN approach.

Finally, Chapter 6 summarizes this thesis' contribution and outlines future research directions.

Chapter 2

Team-centric Asset Sharing in Collaborative Environments

2.1 Introduction

The management of smart environment assets in multi-partner collaboration scenarios, where different sets of assets are owned and operated by different partners, is a non-trivial problem due to the highly interacting and interrelating entities that exist in them and different asset sharing policies applied by collaborating partners.

This piece of research work formalizes, evaluates and compares two asset sharing models, investigating their impact on MSTA-P, a policy-enabled version of an existing asset-task assignment protocol. The first sharing policy, is based on a traditional asset ownership model where assets belong to a collaborative partner and may or may not be shared with other partners, and the second is based on an edge, team-centric model, where users are grouped into cross-partner teams and as team members they have access to assets belonging to all the partners participating in team. We find that the traditional ownership model allows slightly better performance. However, the novel, team-centric approach, presents a more focused sharing model, providing access to the most qualified users, and has a low-overhead when applied in highly dynamic, multi-partner environments, where asset sharing policies need to change frequently. We conclude that team-centric sharing model offers a viable alternative sharing approach.

2.2 Background & Related Work

Organizations in collaborative operations commonly work in peer-to-peer formations and act as servers and/or clients providing and/or consuming information resources provided by others. In order for a coalition to operate effectively, it is necessary for information to move across the organizations' boundaries in an efficient and secure manner [73]. Thus, coalition partners need to develop a number of constraints, which regulate access control on their resources in order to build trust and confidence among the coalition and establish smooth collaboration. The scenarios that we consider are highly dynamic therefore, collaborating partners need to be able to share their resources in a dynamic manner. Partners can be periodically involved in an operation, while it is possible for the same partners to be adversaries in another. The sharing restrictions are expressed by policies enforced through PBMS.

Some of the most well known approaches for asset sharing in multi-organisational environments are represented in [34, 36, 54, 68]. In particular, [34] proposes a sharing model where new collaborative members can only have access to a specific resource if they are first invited by an authorized asset owner partner. Authors in [68], propose an asset sharing approach based on user profile model. This role-based framework combines users' characteristics such user IP address with other environmental parameters such as spatiotemporal data in order to profile a user and decide whether to allow or deny access to the owning resources. Work in [54], proposes an access control mechanism based on a concept-level semantic model, which grants access to different resources to requesting users considering different sets of semantic relationships between owner and requestor, supported by an ontology. The proposed model first identifies the relationships among concepts, then categorizes them and proposes sharing policies based on these categories of relationships. The approach proposed in [36] is based on automated trust negotiation model. In particular the authors focus on a type of contract in which collaborating partners mutually agree to make available some amount of specified resource over a given time period.

In scenarios that this chapter 2 deals with, mobile, multi-agent entities are involved. Thus, the modality with which the entities/users may move on the operational field, can be a factor that affects the asset sharing models' behavior, so we believe it needs to be investigated too. Several sophisticated mobility models associated with scenarios we study have been proposed in the literature. For example, [57] analyses a model where the users traverse an unsecured area along the same path one team at a time until the frontal team arrives somewhere secured. Authors in [62] instead, propose a mobility modeling framework based on fundamental environmental factors such as targets, obstacles and dynamic events occurrence in it. We test our sharing policies by applying and comparing two mobility models as explained in detail in following section named *Moving user model* and *Moving team model*.

Although all the above asset sharing models cope with the problem of resources and services sharing among collaborating organizations in a secure and confidential manner, they do not address directly highly dynamic environments. Thus, they are likely to fail or encounter difficulties in being applied to those cases. In order for all of them to comply with the situational changes, an extra overhead is needed due to the fact that the decision making centre – and therefore, the policy making centre – is far away, (e.g. in terms of spatial or chain of command distance) from where the changes take place. Differently, the team-centric sharing model[1] proposed in this thesis does not present any overhead regardless the frequency of the environmental changes. The teams on the edge of the network – being event-driven entities – are formed, reformed or disassembled as a response to situational changes. Therefore, sharing policies based on a team-centric model are always up-to-date to the unfolding operations.

[1]In this thesis we use the terms sharing policy and sharing model interchangeably

2.3 Asset Sharing Policy Models

In this chapter we present the novel, team-centric model; a sharing model suitable for high-level, policy-based, asset sharing in dynamic and collaborative environments and we compare it with the traditional asset ownership approach under the spectrum of a smart environment asset sharing scenario. The asset sharing policies that we formalize and compare in this work are binary; that is, they either give full or no access to services provided by assets unlike non-binary ones, which using techniques such as obfuscation [16], can grant access to subsets of assets' available services/capabilties.

The first asset sharing policy model is based on the traditional ownership approach. It considers a model making resources either available for any involving partner to use or alternatively reserving them for the exclusive use of the owning partner. We experiment with different sharing levels by allowing collaborative organizations to share different proportions of the assets they own. We refer to this policy model as *Asset-centric sharing*.

In a typical multi-partner operations usually there are number of smaller, more focused formations, which are dynamically created in response to an on the field event and execute concurrent missions for only a short time [75]. In the second sharing model, we assume collaborative partners sharing none of their owning resources "by default" – that is we abandon the asset-centric sharing model. Instead, we introduce a mechanism of cross-partner formations (i.e., small, focused formations), which we call *teams*. Following the Edge C2 model it allows users who participate in the same team to share assets freely [8]; therefore, team members have access to all assets owned by any organization represented in the team. We call this *Team-centric sharing* policy model and we experiment with a variety of sharing levels by applying different degrees of homogeneous (i.e., teams comprised of members from a single partner) & heterogeneous (i.e., teams comprised of members from two or more partners) teams.

In unexpected scenarios such as humanitarian relief and emergency response operations, two or more organizations that own and operate disparate sets of IT and infrastructure assets, merge forces and form coalitions to achieve common scenario objectives. Below we present a formal representation of the sharing policies using the following predicates. Suppose U, U' are users that operate on the field. A is a smart environment asset deployed on the field, owned by a user. P is a coalition partner (in the context of an emergency response operation the coalition partners can be for instance the Police force and the Red Cross), and T is a team of users (i.e., cross partner, small, focused formations).

canAccess(U, A) == true **iff** U can access asset A
hasPartner(U, P) == true **iff** U belongs to partner P
ownsAsset(P, A) == true **iff** partner P owns asset A
hasTeam(U, T) == true **iff** U is member of team T

Asset-centric sharing policy:
canAccess(U, A) **iff**

hasPartner(U, P) $\wedge$ ownsAsset(P, A)

Team-centric sharing policy:
canAccess(U, A) **iff**

hasTeam(U, T) $\wedge$ hasTeam(U', T) $\wedge$ hasPartner(U', P)

$\wedge$ ownsAsset(P, A)

2.4 MSTA-P Protocol

This section describes the MSTA protocol and presents the new, policy-regulated version of it named MSTA-P. We also define the variables we consider for the evaluation

Algorithm 2.1: Initial Negotiation

```
1: for all $A$ within $SR$ do
2:     if canServe(A, T) then
3:         addBundle($B_{AT}$)
4:         calculateUtility($B_{AT}$)
5: distributeBundle($B_{AT}$)
```

and comparison of the formalized sharing models measuring the impact they have on the MSTA-P protocol's performance. MSTA as presented in [77], is a distributed protocol for automated allocation of sensors to the tasks they best serve, considering the task information requirements and the sensor capabilities. As it was mentioned in introduction, while the MSTA protocol is designed for sensors-tasks allocation, we claim that it can be applied to general services provided by any smart environment assets, if one considers sensors bundles as sets of composite, asset provided, sensing services. For that reason herein, we assume a broader MSTA's application domain, considering it as a protocol not just for sensor-task, but general asset-task allocation.

The initial MSTA distributed protocol runs on two main entities; a) the user devices (e.g. smart phone or tablet) and b) the smart environment assets and it consists of two main stages:

- *The initial negotiation stage:* the user devices, respond to user generated tasks requests, compute the best set of assets which may satisfy the request, and distribute the generated bids to this optimal set of asset bundle.
- *The bundle formation stage:* the assets decide upon which bundle to join in order to serve a particular task, giving priority to the most important tasks to which they can provide the highest average utility.

Algorithm 2.1 performs the steps of MSTA *initial negotiation stage*. When a user creates a new task their device queries the assets within a range of the geographical area

Algorithm 2.2: Bundle Formation

```
1: sort($B_{AT}$, $L[n]$)
2: for all $A$ of bundle $B$ do
3:    if isFree(A) then
4:       accept($L[B_1]$)
5:    else
6:       if calculateUtility($B_1$) $>>$ calculateUtility($B_{current}$) then
7:          accept($L[B_1]$)
8:       else
9:          return busy
```

of interest asking for their characteristics (i.e., availability status, location and type). If the assets' *A* capabilities can serve the task's *T* requirements, and their sensing range *SR* covers the area of interest, it adds the asset in the assets bundle B_{AT} and calculates bundle's joint utility (i.e., how well a asset bundle can serve the task); this pair *{asset bundle and joint utility value}* is called a *Bid*. Finally, the device distributes the bid to all the assets involved in the bundle.

Algorithm 2.2 performs the steps of MSTA *bundle formation stage*. At this stage each asset node keeps a list *L[n]* of bids in which it is involved. The list is sorted by decreasing average contribution, meaning the bundle utility divided by number of assets composing the bundle. If an asset is currently not serving any task (i.e., it is free) then it accepts to serve the first bid of the list. Otherwise, if it is currently allocated to a task it will only be preempted from the current and accept serving the new one if the contribution to the new task is strictly greater than the current one (i.e., $utility_{new} >> utility_{current}$). If any one of the assets in the bundle does not accept to serve the task then a new bid (i.e new pair of assets bundle, joint utility) is created and distributed by the initial negotiation stage Algorithm 2.1.

Each task in the protocol, amongst other features (task priority, utility demand and area

Algorithm 2.3: MSTA-P

```
1: for all $A$ within $SR$ do
2:     if canAccess(U, A) then
3:         addCandidateAsset(A)
4: for all candidateAsset[A] do
5:     if canServe(A, T) then
6:         addBundle($B_{AT}$)
7:         calculateUtility($B_{AT}$)
8: distributeBundle($B_{AT}$)
```

of interest) is characterized by two additional variables. The expiration time, which is a deadline within which the task must be supported by an assets bundle or alternatively must be dropped and the duration time, which defines the time during which the task remains active on the field. Each task's expiration time is equal to half of its duration time, which is randomly chosen between 10 and 20 timesteps. Given that the available resources are scarce, we assume that a subset of created tasks will not be supported, which implies a set of dropped tasks. A task is considered dropped (i.e., unsupported by the smart environment network), if there are no available resources to satisfy the task's utility demand in the initial negotiation stage, or if no resource can provide support to the task on time, during the bundle formation stage. We refer to the set of these tasks as *dropped tasks*.

In order to make the MSTA protocol policy-enabled and integrate different sharing policies within it, we need to modify its initial negotiation stage. Thus, we propose the MSTA-P protocol as presented in Algorithm 2.3. The policies evaluation is the first process executed in the protocol after the creation of a task. When the users create a new task, their devices query the assets within the area of interest in order to create assets bundle able to serve the task. The sharing policies are evaluated at this step taking into account if the tasks' creator users can access a specific asset, based on the sharing policies set by coalition partners (i.e., if *canAccess(U, A)* == true). As a result of the policies' evaluation is the creation of a list of assets that could be accessed by the task's creator. We call this set of assets *candidate assets per task*. Therefore, the sharing policies, affect the assets bundles creation by limiting the number of assets a user can access based on the applied sharing policies. The second stage of the protocol is executed as explained before.

In next section we use these two variables, the *candidate assets per task* and the *dropped tasks %* as indicators of the protocol's performance, which is used as a metric for the evaluation of the proposed sharing policies.

2.5 Policies Evaluation & Comparison

For the implementation of MSTA-P protocol and the evaluation and comparison of the formalized asset sharing models we use REPAST Symphony[2], an open source agent-based and discrete time simulation environment. We use the simulation platform to simulate a smart environment network, composed of heterogeneous static “smart" infrastructures and mobile users. We keep the task creation (i.e., users demand for asset services) rate stable at arrival rate equal to 1 task per timestep and we repeat each simulation 10 times for 10000 timesteps averaging the measurements.

Implementing a multi-partner operation scenario we assume 2 partners (partner A and partner B), 250 static assets and 50 mobile user nodes, which are randomly deployed on a 2D grid of 500m x 500m. The users are equally distributed to each partner (partner A = 25 users, partner B = 25 users). In the experiments where the team-centric sharing model is tested, teams are formed at the first timestep and they are maintained identical until the end without any changes. Each team consists of minimum two members while there is no upper bound on each team's size. They may consist of users of the same or both partners depending on whether a team is homogeneous or heterogeneous (as explained above).

[2]repast.sourceforge.net - last checked September 28th, 2015

All the users are identical (e.g. same capabilities, responsibilities) apart from one user for each team who is the team creator and leader. At the beginning of the simulation (i.e., simulation's first timestep) 25 out of 50 users are randomly selected as candidate team leaders. Each one of them consecutively queries the nearby users within a radius of 100m in order to create a team. We call this radius *Team Range* = 100m (TR). If the queried users are free (i.e., do not belong to any team), the team leader adds them to their team. Each user belongs to maximum 1 team at a time, while there are users who do not belong to any team (see section 5.6.2).

As far as the mobility of the users is concerned, we test our sharing model by applying two models. In the first, the mobile user nodes are free to move with no constraints using a random waypoint mobility model. In this case the mobile nodes move independently of other nodes and we refer to this as Moving User Model. In the second we apply a more realistic mobility approach inspired by the nomadic community model [9], which is usually applied in operational environments like the ones we study. In such cases where teams of users collaborate, chasing common objectives it is likely for the users to follow the team leader (i.e., team creator). Therefore, in this model there is a spatial dependency among node's movement and we refer to this as Moving Team Model.

In order to have a complete picture of the sharing policies' effect on the MSTA-P performance, we experiment in the first sharing model by linearly varying the sharing level, starting from 0% sharing ratio (i.e., minimum sharing level where none of the smart environment assets are shared with other non-owning partners) and we increase this ratio by 25% for each experiment until 100% sharing ratio (i.e., maximum sharing level where partners share all of the smart environment assets they own) is reached. In the second sharing model we experiment with the degree of the homogeneous and heterogeneous teams operating on the field. We start with 0% heterogeneous teams (i.e., minimum sharing level) and we increase the ratio linearly by 25% for each experiment until we reach 100% heterogeneous teams (i.e., maximum sharing level). 0% heterogeneity means that all the teams in the field are homogeneous and 100% heterogeneity means that all the teams are heterogeneous. In essence, by increasing the degree of teams' heterogeneity, indirectly we increase the overall shared assets but unlike the first sharing model we do so through teams.

Comparing the two sharing approaches the minimum possible sharing level of asset-centric sharing model (i.e., partners share none of their owning resources) is equivalent to having 0% heterogeneous teams on the field in team-centric sharing model. Conversely, the maximum sharing level of asset-centric sharing model (i.e., partners share all of their owning resources) is equivalent to having 100% heterogeneous teams in team-centric sharing model.

For the evaluation and comparison of the sharing models considered in MSTA-P, we run five sets of experiments. In the first and second set, presented in Figure 2.2 we experiment with the two different sharing models (asset-centric versus team-centric model) respectively. In both experiments, the 250 asset nodes are equally distributed to the collaborating partners (partner A owns 125 assets, partner B owns 125 assets) and we apply the Moving User Model as the user nodes' mobility model. In all Figures the error bars on clustered column charts represent (+ / -) 1 Standard Deviation

In the third set of experiments, represented by Figure 2.3, we experiment with the team-centric sharing approach and we compare how the 2 different user mobility models (Moving User Model & Moving Team Model) affect its behavior. In the second model the mobile user nodes, members of a team are restricted to move within a radius of TR (100m) from their mobile team leader.

In the fourth set of experiments, represented by Figure 2.4, we focus on the team-centric sharing model and we make three different assumptions in terms of asset ownership proportion. In the first case the resources are equally owned by the partners 50% - 50% (as in previous experiments), in the second case the asset distribution is 25% - 75% and in the third and most extreme case, the distribution is 100% - 0%. The 50% - 50% case means that each of the two partners owns half of the smart environment's assets, while in the 100% - 0% distribution only one of the partners owns all the available resources. We do so in order to measure the team-centric policy behavior in extreme ownership-proportion conditions. User nodes move under Moving User Model.

In the fifth set of experiments, represented by Figure 2.5, we apply again the 50%-50% asset ownership case while the user nodes move under Moving User Model and we measure the average proportion of the “domestic" and “foreign" assets that are overall accessed by users. Domestic assets represent the assets that have the same origin as the user to whom they provide services; foreign assets instead, represent those that are owned by different partner. We use the proportion of the “domestic" and “foreign" assets as a metric for defining the efficiency of the proposed asset sharing models. We assume that the larger the proportion of the foreign assets, the more efficient the sharing model is.

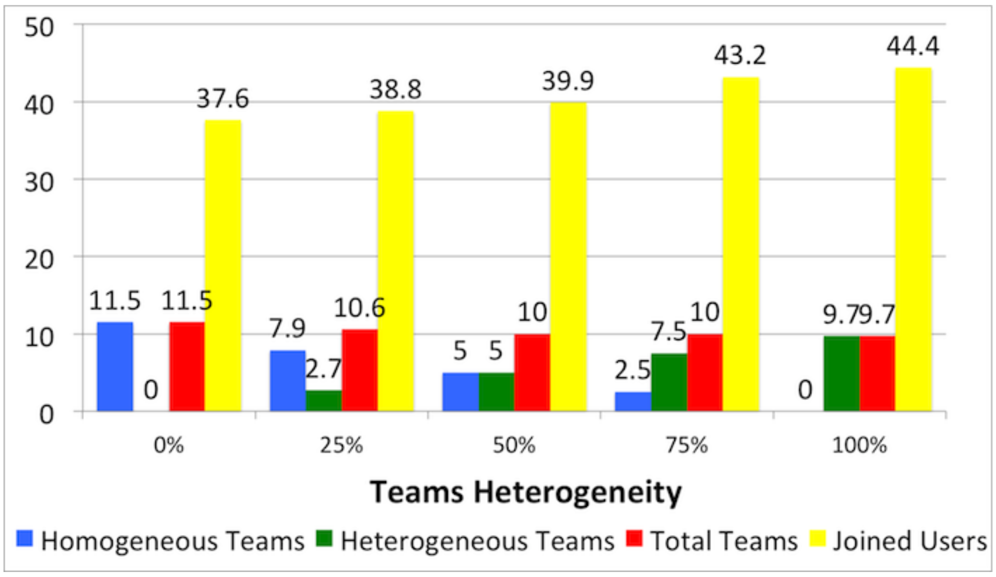


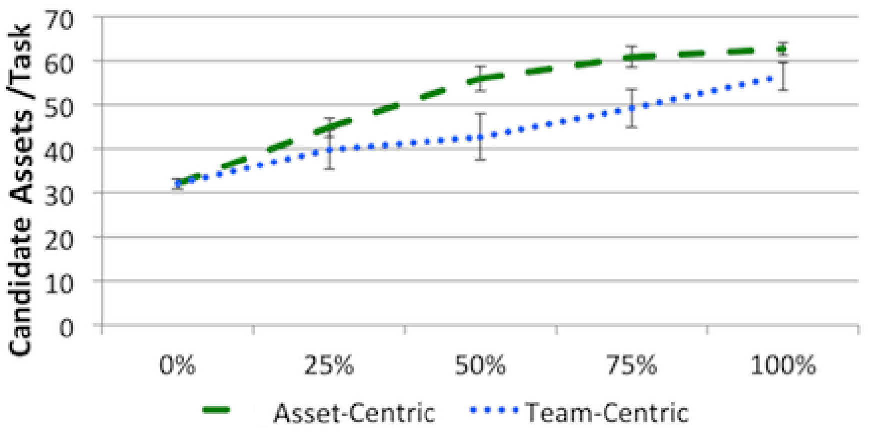
Figure 2.1: Teams statistics: Number of Homogeneous/Heterogeneous teams & teams' Joined Users.

2.6 Simulation Results

Figure 2.1 shows the teams' creation statistics, which are related to all the sets of experiments where teams are involved. We present this figure because the performance of the team-centric sharing model is fully associated with the number of users involved in teams. We note that as the degree of teams' heterogeneity increases, the total number of users involved in teams increases as well. This is due to the restrictive "users from the same partner" condition that applies in the homogeneous teams' formation. As it is shown the number of joined users is ~ 38 for 0% heterogeneous teams and ~ 44 for 100% heterogeneous teams while the overall average proportion of users joined to teams is near 80% of the total users. Additionally, the average number of users per team (i.e., $\frac{JoinedUsers}{TotalTeams}$) increases, while the degree of teams heterogeneity increases as well. Thus, the average number of users per team is 3.3 for 0% and 4.6 for 100% teams heterogeneity.

Figure 2.2 shows the comparison of the two sharing policy models (see experiment set 1 & 2 in Section 2.5). In this experiment we measure the MSTA-P protocol performance under different sharing ratios applied on the asset-centric model and different degrees of homogeneous-heterogeneous teams in team-centric model. Both of them have the same starting point because we start with the minimum sharing level (0% asset sharing ratio & 0% teams heterogeneity).

More in details in asset-centric approach when the sharing ratio is at its maximum (i.e., 100%), the number of candidate assets per task is twice as much as when the sharing ratio is at its minimum level (i.e., 0%). The difference in dropped tasks proportion is even larger. The total dropped tasks when the sharing ratio is at its maximum are almost 8 times smaller compared to when the sharing ratio is at its minimum. Moreover, we observe that the difference of candidate assets per task and dropped task proportion moving from 75% to 100% assets sharing ratios is significantly smaller compared to when we move to higher sharing ratios at lower sharing levels (e.g. from 0% to 25%). The margin between 75% and 100% is 2 assets per task and 3% units in dropped tasks,



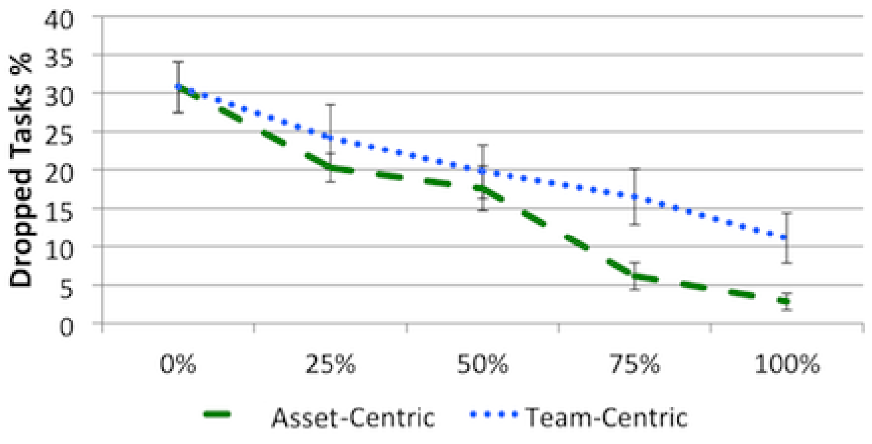
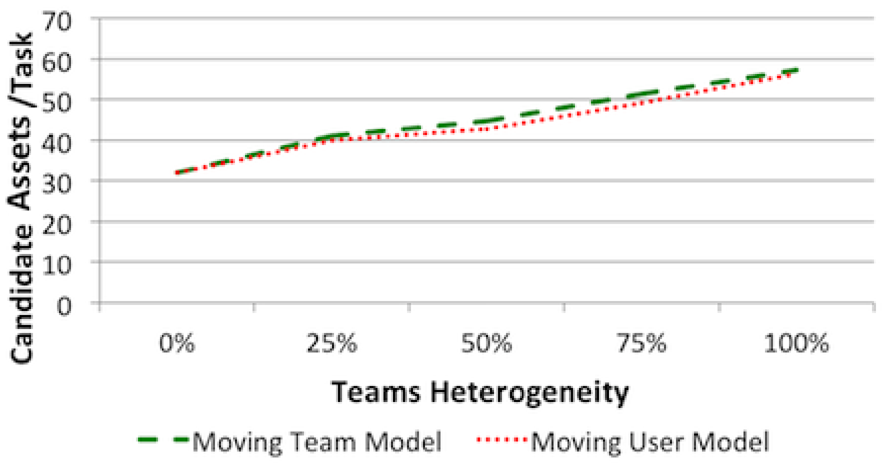


Figure 2.2: Asset-centric vs. team-centric sharing models: effect on candidate assets per task and dropped tasks %.

while the margin moving from 0% to 25% sharing ratios is 11 assets per task and 10% units in dropped tasks. Therefore, the protocol seems to perform better, with short variation, in asset-centric approach for sharing ratios larger than 75%.

As for the team-centric model, in 100% heterogeneous teams case, the number of candidate assets per task almost doubles and the dropped tasks is three times smaller in comparison to when the degree of team heterogeneity is 0%. Moreover, in the team-centric sharing model we obtain that by increasing the team heterogeneity ratio (i.e., increasing the sharing level), we observe a quasi-linear increase in the number of candidate assets per tasks and symmetrically a quasi-linear decrease in the number of dropped tasks, as opposed to asset-centric approach, which follows more of a quasi-logarithmic pattern when increasing linearly the sharing ratio.



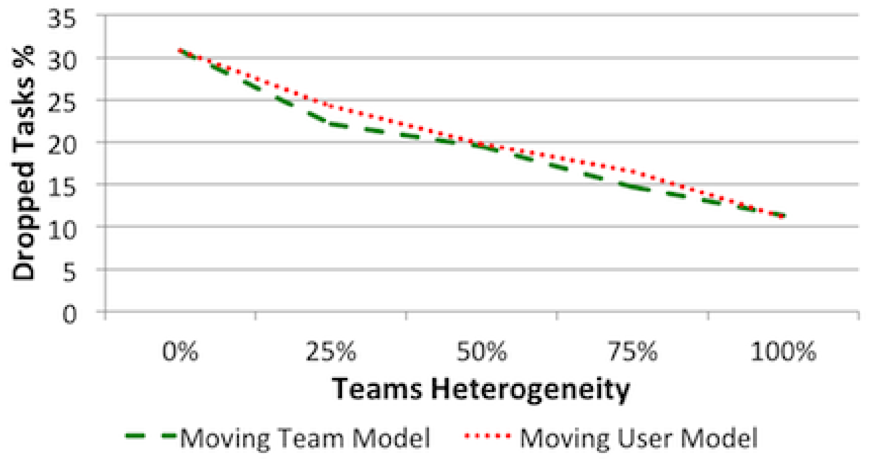


Figure 2.3: Team-centric sharing model: effects of different Mobility models

Overall, the first approach seems to be more effective than the second one, especially as far as the dropped task is concerned. In fact the proportion of the dropped tasks, when the level of both sharing policies reaches its maximum, is less than 5% in the asset-centric approach and more than 10% in the team-centric approach. This is due to the fact that only an average of 80% of the users belong to teams. The 80% team membership implies that another 20% of the users is not involved into teams, given the team formation mechanism explained in simulation setup. This in practice, means that there is a portion of users (i.e., 20% of the total number) that does not benefit from the team-centric model's sharing capability. Thus the impact on dropped task proportion.

In Figure 2.3 we compare the MSTA-P protocol performance with two different mobility models when adopting the team-centric sharing model (see experiment set 3 in Section 2.5)[3]. The results indicate that the protocol behaves slightly better when the Moving Team Model is applied, displaying for each of the different team heterogeneity degrees an average of 1 additional available candidate asset per task and 2% less dropped tasks, compared to the unconstrained mobility model. Therefore, the Moving Team Model seems to be a more effective mobility model in multi-partner operations compared to the random waypoint Moving User mobility model.

In Figure 2.4 we test how the team-centric sharing model copes with different and more extreme ownership proportions (see experiment set 4 in Section 2.5). In each of the three ownership cases the proportion of the assets that can be accessed by users on the field is stable at 50% of the total when 0% asset sharing ratio is applied. For this reason by increasing the degree of teams' heterogeneity the candidate assets per task variable is almost equal for each one of the ownership cases following a quasi-linear increase pattern. As for the dropped tasks % variable, when the degree of team heterogeneity is at its minimum (i.e., 0%), the margin of dropped tasks among different asset ownership proportions is very large ($\sim$10). By increasing the team heterogeneity, the dropped task % in all ownership cases tend to the same point ($\sim$13%). This means that the protocol is less affected by the resource ownership inequalities when the degree of team heterogeneity is high.

Finally, Figure 2.5 shows the average proportion of “domestic" and “foreign" assets that are accessed by users (see experiment set 5 in Section 2.5). The graph on the top presents the accessed assets' origin in asset-centric sharing model when we increase the assets sharing ratio and the one on the bottom presents the accessed assets' origin in the team-centric model when we increase the degree of team's heterogeneity. Both sharing

[3]We do not use ErrorBars in Fig. 2.3 and Fig. 2.4 due to the small margin between the results

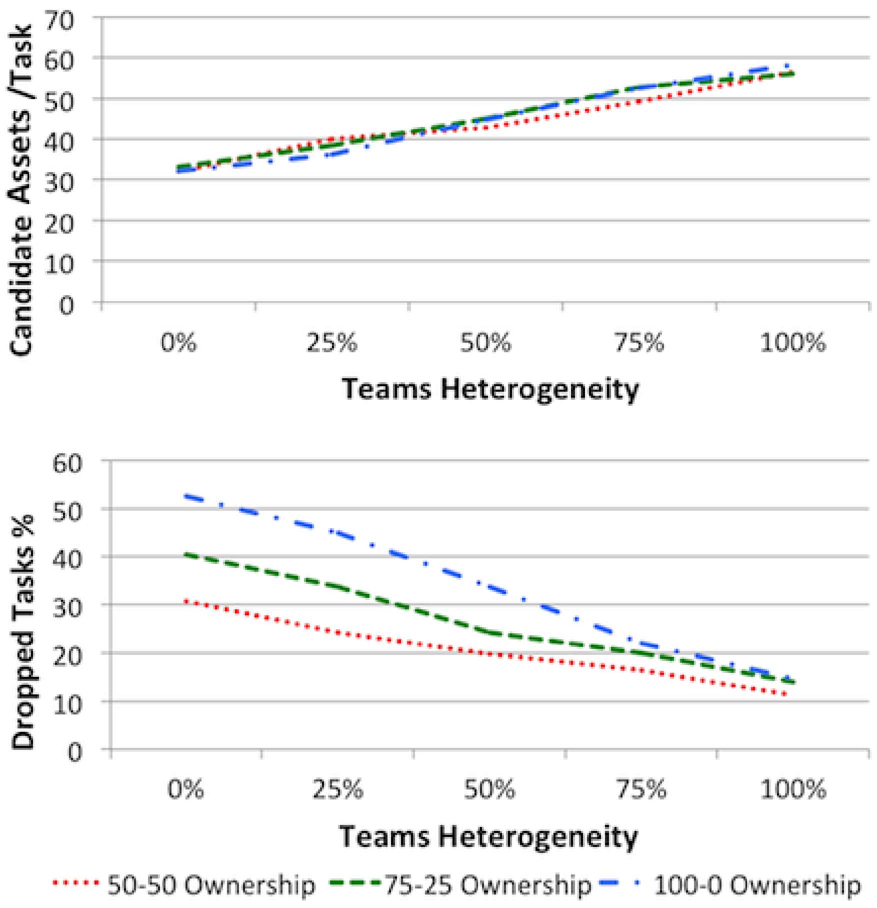
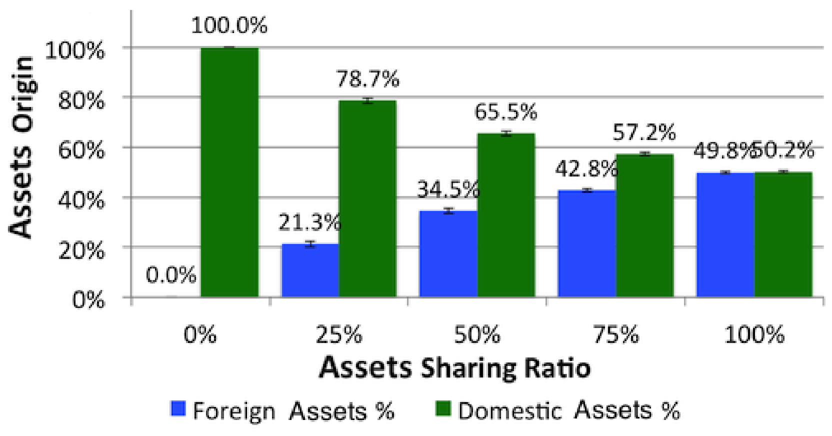


Figure 2.4: Team-centric sharing model: effects of different Ownership proportions.

models start with 100% domestic assets; this is in-line with our expectations because we apply the minimum sharing level. Once again, the asset-centric model seems to perform slightly better than the team-centric model in terms of foreign assets in use. At 25% sharing level, the margin between the two sharing models is 2 percentage units and it increases by 1 unit while we increase the sharing levels and reaches 5 percentage units at maximum sharing level.



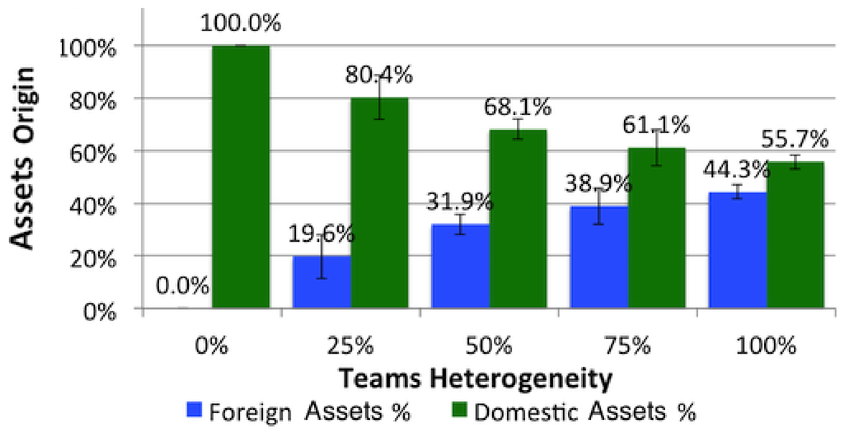


Figure 2.5: Asset-centric vs. Team-centric sharing models: effect on assets origin

2.7 Discussion & Conclusion

Although the asset-centric model performs slightly better according to the simulation results, there are additional reasons that might make the team-centric policy a more attractive option for multi-partner asset sharing in smart environments. Firstly, as stated in [44] the Edge C2 outperforms traditional approaches as well as better supporting coalition-style operations which are increasingly the norm.

Moreover, through asset-centric approach the collaborating partners can only share their assets at coalition level while the team-centric approach is a more flexible sharing strategy giving them the ability to share their assets at a team level. Suppose the aforementioned simulated scenario where there are two partners: partner A and partner B. When partner A decides to share a resource with partner B using the asset-centric approach, all the users belonging to partner B would be able to access this resource. Of course the ownership-based sharing model can achieve more fine-grained access control all the way down to an individual user level. However, this is a highly complicated task dealing with dynamic environments even for role-based sharing approaches such as the one discussed in [68] and this is because users can be periodically involved in an operation, while it is possible for the same users to be adversaries in another.

Differently, in team-centric model, only a specific proportion of partner B users (i.e., those that belong to the same team in which users from partner A are involved) will be able to access it. Thus, we decrease the overall number of users that can eventually access a resource, increasing the level of partners' privacy. Furthermore, while the team-centric approach decreases the number of users that can access an asset, it also increases the “quality" of users that can access it. In fact, as mentioned before teams are entities in which users are all focused towards a narrow specific objective; thus, sharing resources with users of the same team is usually beneficial provided that the team members will be able to better carry on their work towards a common goal, and that they will most likely have complementary expertise. Users who are involved in teams are in fact the subset of users that can contribute the most to the task and hence, to the coalition. It is therefore very reasonable to allow them to access assets owned by partners represented in the same team.

We conclude that: (a) the asset-centric does not outperform the team-centric model's efficiency with a large margin, (b) the team-centric model presents a more focused sharing approach providing higher privacy by restricting the asset sharing in multi-partner operations and it also filters the accessing users by providing access to the most qualified ones, and (c) the team-centric model is based on an edge event-driven and low-overhead sharing formation mechanism compared to asset-centric. Thus, the team-centric model seems to be a viable alternative asset sharing approach for highly dynamic multi-partner operations.

Chapter 3

Controlled Natural Language Policy Representation

3.1 Introduction

In this chapter we present the CE policy language. CE representation provides the enabling technology for the proposal of a policy conflict analysis mechanism and a novel interest-based policy negotiation mechanism presented in Chapters 5 & 6 respectively. CE policy language is a general purpose language aiming to bridge the gap between non-IT expert policy makers and PBMS technical complexity in terms of policy rules' development. It is based on Controlled English which is a type of CNL designed to be readable by native English speakers, whilst presenting information in a structured and unambiguous way, making it machine processable.

In collaborative domains such as military operations, a suitable paradigm for dealing with the management of Intelligence, Surveillance and Reconnaissance (ISR) assets among different coalition partners is the PBMS. Traditionally, policies are created at the center of a coalition's network by high-level decision makers and expressed in low-level policy languages such as Ponder or Common Information Model SPL by technical personnel, which makes them difficult to be understood by usually non-technical users operating at or near the edge of operations.

Moreover, policies must often be modified typically in rapid response to operational changes. Commonly, the users who must cope first with situational changes are those at the organization's edge, so it would be effective if they were able to write new, update or delete existing policies themselves. We investigate the use of Controlled English as a means to define a human-friendly policy representation. We show how a CE-based model can capture a variety of policy types. The use of CE is intended to benefit coalition operations, by bridging the gap between technical and non-technical users. Although, we do not evaluate the CE language's user-friendliness as a policy representation herein, there is some prior work (e.g., a study with undergraduate students that utilized CE for describing flickr images), that builds an evidence base for the general usability of CE language [79].

3.2 Background & Related Work

The cornerstone of any PBMS are the policy rules. A plethora of policy languages have been proposed the last two decades for the description and development of policy rules. The extensive review of those policy languages is outside the scope of this thesis, given that the vast majority of them were highly purpose-driven representations, and proposed for the development of specific policy-based management systems. In this subsection instead, we make a retrospection and present some of them that took significant attention by the policy-based management community and were focused on the management of large domains of IT infrastructures.

The Policy Description Language (PDL) [55], is one of the first policy languages developed by Bell Labs in late '90s. It was proposed mostly for network management and follows the event – condition – action syntax[1] . PDL has a flat structure without any hierarchical grouping of its rules such as roles or any other concept.

Ponder, maybe the most famous and cited policy language, is a declarative and object-oriented policy language proposed by Damianou et al. from Imperial College [24]. It

[1]Policy's language syntax is explained in Chapter 3

was developed as a language for specifying security and general networking and systems management. Ponder policy language has a role-based representation. Claiming flexibility and reusability purposes it utilizes a structure of highly grouped policies into roles. There are five types of policies in Ponder: a) Authorization policies, b) Filter Policies, c) Refrain policies, d) Delegation policies, and e) Obligation policies, where the first four are used for access control management while the latter following the event – condition – action syntax targets the definition of obligation policies.

Simplified Policy Language for CIM (CIM-SPL) is the policy language proposed by DMTF to complement specifically the policy model included in CIM (Common Information Model) standards of Distributed Management Task Force (DMTF) [3]. CIM is an object-oriented information model representation capable of describing general IT components in domains such as networking and service management [1]. CIM-SPL is expressed through condition(s) – action(s) pairs and given its special purpose nature, it is fully incorporated into the CIM data structures.

Knowledge Acquisition in Automated Specification (KAOS) is yet another famous policy representation used for the development of a management tool for governing software agent behavior, especially in grid computing applications using an ontological representation encoded in OWL [26]. A major distinction of KAOS compared to all of its predecessor policy languages is that it is a goal policy language, not an action one. Thus, it is a good fit representation for expressing mainly higher-level systems' objectives rather than lower-level, action-oriented descriptions.

eXtensible Access Control Markup Language (XACML) is maybe the most well-known representation for the description of authorization policies. It is the OASIS standard access control policy language for web services and widely adopted by both industrial and academic policy-based management communities [69]. It is a declarative language developed based in XML with its latest version 3.0 released in 2013. It is hard to categorize XACML due to its monolithic nature, given that it has been proposed as a solid access control management representation. It is a static representation triggered by a single event.

Finally, Rei is a declarative policy language based on deontic logic developed using OWL-Lite [47]. It is capable of expressing policies of both major types i.e., authorization and obligation. It has been mainly proposed for security and privacy issues management in dynamic computing environments and as a role-based policy, it offers a policies grouping representation.

The CE policy language is based on Controlled English, which is a type of CNL designed to be readable by a native English speaker whilst representing information in a structured and unambiguous way making it machine processable. The near to natural language representation of CE, which implies an easier to learn and use representation makes it a better fit for dynamic and distributed policy-based management especially for non-IT expert policy makers operating near or at the organization's edge. CE and CE policy language are ontological-based models that utilize semantics for representing domains and policy rules that govern them. CE representation is based on first-order logic (FOL); hence, CE policy language can express attribute-based policy rules considering sets of concepts, their attributes and relationships between them. Finally, it can express both, authorization and obligation policies.

3.3 PBMS & Policy Rules

The goal of a PBMS and the role of its inherent policies is to provide a middleware for high level systems management. The policies, depending on the nature of the managed system and the business level objectives of their authors, guide a system's actions basically in regards with security, privacy and performance issues, with security purpose holding the lion's share. Policies that describe systems' security behavior for instance, typically aim to manage the system such that it complies with specific security standards [89], while on the other hand, operational policies that are in charge of managing a mesh network of ISR assets, might regulate networking performance issues configuring several quality of service variables [67].

In general, we see the management role of a PBMS and therefore its inherent policies as the guide of system's actions, in order to achieve behaviors that would secure optimal system's outcomes. Any policy definition language should be reasonably human-friendly in principle in order to ease policy makers in the task of policy rules creation and maintenance. On the other hand, given that this representation needs to be processable by a diverse set of devices (with different processing capabilities, interfaces etc.) that act as policy decision points, it should be also machine-processable [1].

The policy-based system administrators should be able to perform three operations on policies; they should be able to add new policies, update existing ones and finally, dispose those they do not reflect their goals any more. Each policy, right at its point of entrance to the system, should be analyzed for syntactic correctness. Often, in the proposed PBMS the policies are usually translated from a human friendly representation, used by the authors for interacting with the PBMS through some kind of graphical user interface (GUI), to a formal representation amenable to analysis by an inference engine.

Regardless of the goals that reflect and the systems that they intent to manage, policies can be classified into two major categories:

- Authorization policies
- Obligation policies

The first describes what actions the subject of a policy is permitted (i.e., positive authorization) or forbidden (i.e., negative authorization) to do to a set of policy targets. Authorization policies are the cornerstone of access control systems. The obligation policies on the other hand, describe the actions the subject of a policy must (i.e., positive obligation) or must not do (i.e., negative obligation) to a set of policy targets [58].

An additional feature for the categorization of policy languages is their syntax. In the literature there are two major policy syntax trends; the subject, action, target (SAT) and the event, condition, action (ECA) [10, 24] models. In the SAT case the Subject specifies the entities (human/machine) which interpret obligation policies or can access resources in authorization policies. The Action determines what must be performed for obligations and what is permitted for authorization policies. Target describes the objects on which actions are to be performed while the optional Constraints are boolean conditions, which when they occur, the policy is applicable. ECA model reads as follows: when an Event occurs if Condition(s) is/(are) true then execute Action. The Condition and Action objects are similar to SAT model which are also event-triggered, although lacking an explicit Event element in their declaration. Thus, the difference between the two lies in Event object which is embedded in the ECA model's body [1].

The IBM Controlled Natural Language Processing Environment (CE Store)[2] software provides an environment where CE can be used to model a domain and reason about it. The parser, the syntax analyzer of CE as well as its reasoner, are all integrated into this software platform, which is utilized as an out of the box tool herein, playing the role of the GUI between, policy authors and PBMS. CE Store is where the syntactic correctness of CE policies is checked, since the policy rules follow the prescript of CE syntactic rules. Each policy rule represented in CE policy language follows the if – condition(s) – action form and consists of four basic grammatical blocks a) **Subject:** specifies the entities (human/machine) which interpret obligation policies or can access resources in authorization policies, b) **Action:** what must be performed for obligations and what is permitted for authorization, c)**Target:** objects on which actions are to be performed and d) **Constraints:** boolean conditions. With CE languages we can express policies for both major categories (i.e., authorization and obligation) and it follows the SAT syntactic model.

[2]https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=558d55b6-78b6-43e6-9c14-0792481e4532

3.4 Achieving Edge C2 Utilizing CE

In this section we present an overview of Edge C2 [8] approach, we discuss its strengths and characteristics and we also investigate in which military operations it better fits based on the literature. Moreover, we discuss the benefits of an Edge C2 that utilizes CE language as policy representation by highlighting the advantages of CE in terms of user-friendliness and understandability as well as how it benefits non-technical policy makers at the edge.

As noted in [88] the complexity, uncertainty and dynamics of military operations, as far as the collection of entities involved in such endeavors, has highly increased in recent years. Therefore, the demand for more agile corresponding forces (e.g. human decision-makers, ISR systems and system management methods) increases as well. Over the years the nature of C2 has significantly changed and the currently applied approaches, which are described and further analyzed in [8], are presented below.

- Conflicted
- De-Conflicted
- Coordinated
- Collaborative
- Edge

Each of these differs from the others along one or more of the following dimensions: (1) Allocation of decision rights, (2) Patterns of interaction and (3) Distribution of information. Moving from the *conflicted* to the Edge C2 approach the allocation of decision rights goes from no rights to the collective, to dynamic and tailored self-allocation to individuals. The patterns of interaction that take place between and among entities goes from no interaction to unlimited and unbounded horizontal interaction, while the information distribution goes from crucial organic information to real time sharing of all available and relevant information (in accordance with policies).

As mentioned previously, the Edge C2 model empowers the users who operate at the edge of the network while in addition allows for intra-edge communication without requiring permission from a central authority. Crucial preconditions for a successful application of the Edge C2 model apart from the need for enhanced peer-to-peer horizontal interaction among the users on the field, is the moving of senior personnel into roles operating at the edge [43]. As a result, the need for intermediaries is reduced and an unbundled C2 is achieved. Thus, commanders operating at the edge become more responsible and take further substantial initiatives such as, the sharing and allocation of resources and engagement rules establishment in a highly dynamic manner as a response to operational changes.

Controlled Natural Languages were first designed to improve human comprehension, having no specific application domain or narrow community in mind [52]. Thereafter, they were applied to technical documentation for improved human comprehension and have been used for the improvement of machine translation. Finally, in the mid 1990s, many CNLs were proposed providing some sort of mapping to formal logic. They tried to bridge the gap between formal representation languages (e.g., OWL[3]) [85] and natural languages (e.g., English) and introduced a user-friendlier knowledge representation compared to common formal languages, which are admittedly hard to be understood by people unfamiliar with formal notations [93].

CNL being a subset of natural languages (NL) are less complex and ambiguous, so they present improved interpretation for machines compared to NLs. In this work we use Controlled English [66] as a means to define a policy representation. CE is a type of CNL designed to be readable by a native English speaker whilst representing information in a structured and unambiguous way. The structure of CE is simple but

[3]`http://cies.hhu.edu.cn/pweb/~zhuoming/teachings/MOD/N4/Readings/5.3-B1.pdf`

fully defined by a syntax, which makes the language parsable by computer systems. CE is aspiring to provide a human-friendly representation format that is directly targeted to non-technical, domain-specialist users to encourage a richer integration between human and machine reasoning capabilities [66].

We argue that it is in general difficult to measure how easy or difficult it is for a language to be understood and learned by a human. Since we have not experimentally tested CE's understandability with the understandability of other lower level well-known and widely used policy languages such as CIM-SPL we cannot safely claim that CE is a user friendlier representation than its predecessors. However, there are in literature a number of works [14, 42, 51] that conducted experiments in order to test and compare the user-friendliness of CNL representations to the user-friendliness of formal languages such as OWL (not in terms of policies development). The results of those experiments in all cases led to the conclusion that CNL can do better in terms of understandability compared to formal languages; in addition they can achieve significantly better results in situations where users have little or no technical training.

We introduce and further explain the CE structure and syntax in section 3.5 where we define the coalition assets sharing ontology. Here, we present a simple authorization policy rule expressed in both: CIM-SPL in Table 3.1 and CE in Table 3.2 representations to show the different levels of human-friendliness of the two. Suppose the simple scenario in a coalition operation context where an authorization policy allows a user to access an asset, if the user and the asset are both affiliated with the same partner. Hence, the policy rule's grammatical blocks are as follows:

Subject: user
Target: asset
Condition: user's partner = = asset's partner
Action: allow

```
Condition
{
    subject.affiliation() = = object.affiliation()
}
Decision
{
    canAccess.allow()
}
```

Table 3.1: CIM-SPL representation

```
if
    (the asset A is affiliated to the partner P)
and
    (the user U is affiliated to the partner P)
then
    (the user U canAccess the asset A)
.
```

Table 3.2: CE representation

As it is shown in these example, CE seems to be a more user-oriented representation compared to CIM-SPL, while instead CIM-SPL is more concise compared to CE. It is admittedly straightforward for an English speaker/reader to understand the policy rule expressed in CE even if they have little or no knowledge of the domain model. CE representation is not far from the policy plain-text explanation above. On the contrary, in order for a user to understand the policy rule expressed in CIM-SPL, some technical-programming skills are needed. However, some training is also needed for a user in order to write policy rules in CE; for example, they need to be aware of the underpinning CE model which determines what kind of sentences can be stated in CE.

Given that CE is defined by syntactic rules that are inherently close to NL's rules, we believe that the training time for an untrained user to learn writing policies in CE language is shorter than the time needed for a user to learn how to develop policy rules in representations such as CIM-SPL. Thus, the use of CE as a means for policy-based management of resources in coalition operations seems to be a good alternative in operations where the Edge C2 is applied.

Furthermore, the use of transparent languages like CE, which is understood by a larger number of operators, we believe strengthens the connection between loosely-coupled partners of coalition by helping them to understand each others' approaches, determine their needs, measure their progress and develop common strategic vision [8]. A side effect of CE's usage is that the civilians in the AO might be able to potentially contribute to an operation due to CE's high readability that makes it understood by any native English reader. Thus, utilizing CE can potentially facilitate the establishment of crowd sourcing endeavors.

3.5 System's Conceptualisation and CE

In previous work, Preece et al. used CE to express the elements of an OWL-based knowledge base, which has been developed in order to allow automatic matching of sensing tasks and ISR asset based on the capabilities required by tasks and provided by each sensing assets [39, 80]. The motivation for using CE in that case was to increase the transparency of the system in order to support a more interactive Human-in-the-Loop approach. Making the human-machine interaction more transparent we eased human users to work in a more cooperative manner with the system in exploring the various means for serving their tasks. In this thesis we focus on the development of a CE-based PBMS, extending the previous ontology, in order to deal with the multi-partner, policy-based asset sharing aspect in coalition operations. Assets are owned by a single party and can be either reserved for the exclusive usage by the owning partner or can be shared with different subsets of users affiliated to other collaborators.

Being a type of CNL, CE can be used to define domain models, which take the form of concepts definitions and describe the system to be managed. Obeying to first-order logic these concepts comprise objects, their properties and the relationships between them. CE language supports multiple inheritance and can build hierarchies of concepts; using the "is a" syntax in the conceptualise sentences CE can define concepts, which are specialization and inherit properties of other concepts. Once the CE model is built, then it can be instantiated accordingly, to represent the concepts and relationships defined in the model. CE allows any instance to be asserted as any number of concurrent concepts (e.g. "the user U1 is a private and is an intelligence analysts and is a/an...") [105].

Exploiting the reasoning capability of CE we can develop rules, which follow the "if - condition – action" form, and can be evaluated on the domain model. Both, rules and the domain model are expressed in CE representation; no other formal notations are needed. The product of any rule's evaluation (i.e., CE reasoner execution) is a CE sentence which is automatically pushed to the user and can be used as input for PBMS' policy decision point. CE rule's evaluation creates also a rationale which is a set of reasoning steps, each one of which is defined as a "because" relation amongst the rule's conditions. The reasoning steps that create the rationale follow a backward-chaining interpreter.

In section 3.6 we exploit the rules creation ability of CE in order to define a variety of high-level, attribute-based authorization polices. We need to make it clear here that CE is not based on the idea of verbalising rules, which are already implemented in a formal language. The IBM Controlled Natural Language Processing Environment or briefly CE Store is a web application which provides the information-processing environment within which human and machine agents (i.e., Java coded entities) can develop and interact with existing CE-based conceptual models. Within CE Store different types of agents, amongst other, can develop logical inference rules (i.e., policy rules) and execute them on a pre-developed conceptual model.

Before presenting the CE authorization policy rules we first need to define the domain model, which represents the ISR system that needs to be managed and is deployed in a coalition operation environment. The domain model captures the objects, their properties and the relationships between them. For the domain model's definition, we extend and enrich the ontology developed in [80], with additional objects that are usually involved in coalition military scenarios.

The core concepts in our ontology are the users who are the rules' subjects and the assets which are the targets of the sharing policy rules. Users, given a set of sharing constraints (i.e., policy conditions) either can or cannot access coalition assets. They create sensing tasks (which need to be fed with sensing information provided by assets), can be members of special purpose cross-partner CoIs called teams[4] see Chapter 2 and are affiliated to a single coalition partner. Assets can be accessed by users, they are either sensing devices (e.g. camera, microphone) or a service (e.g. acoustic event detector service, camera sensor recognition service provided by a sensing device) and are owned by a coalition partner. The full assets sharing domain model is presented in Figure 3.1.

Briefly, the asset-task allocation is a two step process and consists of: a) the evaluation of sharing policies and b) the asset-task allocation protocol which have been proposed in [77] and has been explained in Chapter 2. First the user creates a task, which in order to be served requires a set of capabilities provided by deployed assets. An entity named *assetList* is created then, that contains all the candidate assets (i.e., sensors and services) that can serve the task's requirements, while can be also accessed by the task's creator after evaluating the sharing policies. Thereafter, the allocation protocol assigns assets to the tasks they best serve, following quantitative (in respect to the matching intelligence providing by the CE-based knowledge base) and qualitative (assets assigned to the tasks which perform higher utility) criteria.

[4]In a typical edge coalition operation within a CoI usually there are a number of smaller, more focused CoIs which are dynamically created in response to an on the field event and execute concurrent missions for only a short time.

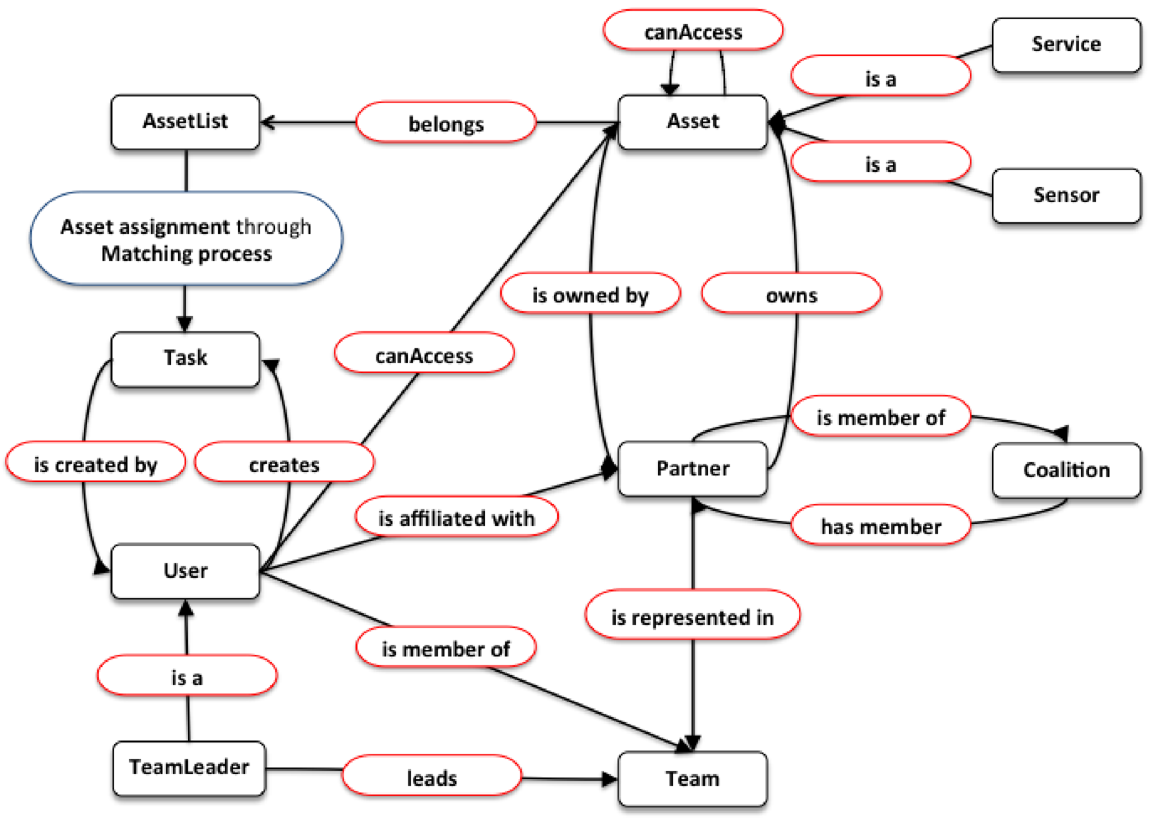


Figure 3.1: Coalition assets sharing ontology.

For simplicity we hide all the complex knowledge base and asset-task matching process into the *Asset assignment through Matching Process* relationship bubble, at the top left hand corner of Figure 3.1. In the domain model the asset owner users can write policies that express both asset sharing models as expressed in previous chapter namely the traditional ownership model and the sharing model based on the edge team-based paradigm.

The following CE definitions cover only a part of the domain model as presented in Figure 3.1. Our intent here is to present the basic capabilities and structure of CE as a domain concept developer.

To create a new object in the domain model we simple conceptualise it as follows:

```
conceptualise the assetList L.
```

For the definition of concepts' properties there are two possible forms, which are semantically identical but allow the subsequent facts to be expressed in slightly different ways:

Verb singular form:

```
conceptualise the team M
  ~ is led by ~ the teamLeader D.
```

Functional noun:

```
conceptualise a ~ coalition ~ C that
  has the partner P as ~ member ~.
```

We can define any number of properties for a concept in a single CE sentence but we currently cannot mix *verb singular* and *functional noun* properties in a single conceptualisation sentence, while we can write as many CE sentences as we like for a single concept. The CE Store will amalgamate all sentences for that concept into the model when it loads the sentences.

To define a property with a *textual value* rather than a relationship to another instance we use the word "value" as below:

```
conceptualise an ~ asset ~ A that
  has the value S as ~ scarcity ~ and
  has the value B as ~ sensorType ~ and
  has the value R as ~ serviceType ~ and
  has the value L as ~ capability ~.
```

The following examples declare that a sensor and a teamLeader are subtypes of pre-existing concepts, asset and user respectively. Sub-concepts have inheritance in the normal way, so if we define a property on a parent concept all children inherit that property.

```
conceptualise a ~ sensor ~ R that
  is an asset.

conceptualise a ~ teamLeader ~ D that
  is a user.
```

The tilde "~" symbol in all the above CE definitions is used by the CE Store's syntax analyzer for the semantic definition of the CE input text. Once the conceptual model is defined, the next step is the model's instantiation through fact sentences. Below we present instantiation examples for some of the objects, properties and relationships represented in domain model of Figure 3.1.

```
there is a partner named UK.
there is a partner named US.

the partner UK
  is represented in the team t1 and
  owns the asset a1 and
  is member of the coalition US-UK.
the partner US
  is represented in the team t2 and
  owns the asset a2 and
  is member of the coalition US-UK.

there is a user named u1 that
```

```
  has 'uid1' as userId and
  has 'intel1' as expertise.
there is a user named u2 that
  has 'uid2' as userId and
  has 'intel3' as expertise.
```

Note that when specifying a value (...has 'intel1' as expertise...) we do not need to say "has the value" but can leave it out for readability. Once we have defined and instantiated the ontology then we can develop and execute rules on it as it is shown in Section 3.6 through the interface provided by CE Store.

3.6 CE Policy Rules

In this section we present the high-level, attribute-based authorization policies expressed in CE. Our goal is to verify whether CE is expressive enough to capture a variety of authorization policy rules or as we call them herein asset sharing rules. We experiment with different levels of resource sharing while showing the flexibility of CE as a sharing policy representation.

Research and development in policy technologies within International Technology Alliance[5] (ITA) project, which is funding the research presented herein, has as consequence the development of PBMS framework named the Policy Management Toolkit [76]. This toolkit was developed to perform a variety of management functions on sets of policies applicable to sensors, sensor platforms, and networks [74]. The developed policies regulate aspects including platform control, sensor and system control, sensor information access control and information flow protection.

In this chapter, utilizing the CE policy language, we focus on authorization policies that regulate users-assets access control in the context of a coalition operation. Such

[5]`https://www.usukitacs.com/`

policies should take into account several factors including user's partner membership, user's membership of cross-partner team and user's echelon and expertise. As far as the assets are concerned authorization policies should take into account factors such as asset's partner ownership, asset's scarcity and intelligence capabilities. In addition, given that the scenarios that we are dealing with are time critical ones, CE-based rules must be able to consider and capture also spatio-temporal parameters in terms of asset sharing.

We present below a number of high-level access control rules using each time as building blocks different attributes of the concepts defined in the ontology of Figure 3.1. Attributes are sets of properties that are used in the domain model definition to describe concepts. Each rule consists of four basic grammatical blocks, the binary Conditions, a Subject which is either a user or an asset (i.e., service) that wants to access an asset; a Target, which is an asset and an Action that the Subject wants to perform on the Target (i.e., can/cannot access). Apart from those four policy grammatical blocks, each policy is described by a unique identifier ID and a binary policy type parameter that determines whether it is an authorization or obligation policy. In order to test the expressive ability of CE as a policy language we develop policies for sharing different sets of assets (e.g. a specific asset to assets owned by a coalition partner) with different sets of users (e.g. a specific user to users who are members of a team) including those that enforce access control following the traditional, asset-based and team-based paradigms as described in the previous chapter.

Suppose U is a user, A is an asset, P is a coalition partner, T is a team and C is an operational coalition. In addition the property B refers to the asset capability, property E defines user expertise and the asset location property C. Those concepts and their properties are connected with relationships as described by the predicates below:

canAccess(U, A) == true if user U can access asset A

isAffiliated(U, P) == true if user U is affiliated with partner P

owns(P, A) == true if partner P owns asset A

member(U, T) == true if user U is member of team T OR

member(P, C) == true if partner P is member of coalition C

isRepresented(P, T) == true if partner P is represented in team T

capability(A, B) == true if asset A has B as capability

expertise(U, E) == true if user U has E as expertise

location(A, C) == true if asset A has C as location.

To help the reader to understand the meaning and intention of each rule we express each one of them (“Rule 1” - “Rule 4”) in three different ways: with a plain text description, with a formal definition using the above predicates and a CE-based representation. We start with simpler rules moving gradually to more complex ones.

Rule 1

If User U is affiliated with partner P they can access the asset A, which is owned by partner P (i.e., traditional asset ownership model).

canAccess(U, A) **iff**

isAffiliated(U, P) $\wedge$ *owns*(P, A)

```
if
  ( the asset A is owned by the partner P ) and
  ( the user U is affiliated with the partner P )
then
  ( the user U canAccess the asset A )
```

```
.
```

Rule 2

If User U is member of team T they can access asset A if it is owned by partners P and partner P is represented in team T (i.e., team-based asset sharing).

canAccess(U, A) **iff**

member(U, T) $\wedge$ *isRepresented*(P, T) $\wedge$ *owns*(P, A)

```
if
  ( the user U is member of the team T ) and
  ( the partner P is represented in the team T ) and
  ( the partner P owns the asset A )
then
  ( the user U canAccess the asset A)
.
```

Rule 3

If User U has 'intel1' as expertise E can access asset A if asset A has 'detect' as capability B.

canAccess(U, A) **iff**

expertise(U, E) $\wedge$ *capability*(A, B)

```
if
  ( the user U has the value EX as expertise ) and
  ( the value EX = 'intel1' ) and
  ( the asset A has the value CP as capability ) and
  ( the value AP = 'detect' )
```

```
then
  ( the user U canAccess the asset A )
.
```

The policy rules above show that the expressiveness of CE representation in developing authorization policy rules, utilizing sets of concepts, their properties and the relationships between them, as they are described in the domain model definition. The limitations of CE policy language's expressivity, are the limitations of CE language in general, whose limitations (CE is based on first-order logic) are set by FOL expressiveness. FOL is a very expressive representation and can formalize several mathematical theories including set theory. Its variables can describe real world things, while it can quantify over them to talk about all or some of them without needing to refer to them explicitly. In essence the abstraction level and the level of detailed description of the managed system as expressed in domain model's definition, set the limits of CE's expressiveness as a policy language and this in a user-friendly manner. The more detailed the managed system is described, the more complex/fine-grained access control rules one can build. With that being said, with CE we can also express and evaluate policies considering parameters related to temporal and territorial dimensions as shown in "Rule 4".

Rule 4

If User U is member of team T can access asset A if asset A is owned by partners P and partner P is represented in team T and asset A is located in 'Kunar'.

canAccess(U, A) **iff**

member(U, T) $\wedge$ *isRepresented*(P, T) $\wedge$ *owns*(P, A) $\wedge$ *location*(A, C)

```
if
```

```
  ( the user U is member of the team T ) and
  ( the partner P is represented in the team T ) and
  ( the partner P owns the asset A ) and
  ( the asset A has the value L as location ) and
  ( the value L = 'Kunar' )
then
  ( the user U canAccess the asset A)
.
```

In Section 3.5 we mentioned that when a CE rule is executed, a set of sentences which is called rationale is automatically created and can be pushed to the policy makers in order to enhance transparency and understandability regarding the policy rule. Below we present the created rationale after evaluating the Rule 5 on the domain model.

Rule 5

```
if
  ( the asset A is owned by the partner 'US' ) and
  ( the user U has the value EX as expertise ) and
  ( the value EX = 'intel1' )
then
  ( the user U canAccess the asset A )
.
```

Rationale

```
the user 'u1'
canAccess  the asset 'a2'
because
the constant 'intel1' = 'intel1' and
the user 'u1' has 'intel1' as expertise and
the asset 'a2' is owned by the partner 'US'
```

3.7 Conclusion

In this Chapter we utilized a CNL named CE as a means to define a transparent policy representation, which is human-friendly and directly processable by machines. First we defined a CE-based conceptual model to capture the entities of a multi-partner military inspired collaborating operation and then developed and executed on this conceptual model a variety of authorization attribute-based policy rules expressed in CE policy language. We showed that the limits of expressiveness and flexibility of CE as a policy representation are simply defined by the entities of the conceptual model, their properties, and the relationships between them. In the following two chapters we utilize CE and some tools provided by it, such as, its semantics, and its ability to build hierarchies of concepts supporting multiple inheritance, to propose a novel policy conflict analysis and an interest-based policy negotiation mechanism.

Chapter 4

Policy Conflict Analysis in Collaborative Environments

4.1 Introduction

One of the reasons why PBMS are not broadly adopted is that it is difficult to develop and maintain them, whilst guaranteeing integrity by means of avoiding policy conflicts [19]. Policy conflicts occur when two or more policies are applied simultaneously and their resulting actions contradict with each other, leading to unpredictable system behavior. We present a policy conflict analysis model for detecting static and dynamic conflicts using a Controlled English based approach. The model focuses on highly dynamic, multi-partner environments, where the likelihood of conflicting policies is high. The experimental evaluation of its execution, time complexity shows that the CE-based analysis model scales better as the managed system's entities increase, compared to the naive pairwise approaches.

There are two major components any policy conflict analysis tool consists of: a) formal representations to describe the system to be managed (concepts and relationships amongst them) and the policies that govern it and b) an inference engine that is performed on those formalisms in order to determine potentially conflicting relationships between policies [19]. Throughout the existing literature there are two main approaches proposed as far as the first component is concerned. The first utilizes logic-based formalizations for the description of the managed system, such as the event calculus [12], while the second approach utilizes information models and/or ontological representations such as DEN-ng and OWL [87, 90]. Our approach for policy conflict detection is an ontological one as stated in the introduction.

4.2 Background & Related Work

Trying to widen the use of PBMS and make its broad application in system's management a reality, a plethora of policy conflict analysis mechanisms have been proposed throughout the last two decades. We present here some of the most well-known ones that also represent groups of approaches, towards achieving conflict free PBMS.

The work in [28] and [30] complement each other and consider the use of a system information model, specified in UML, for modeling the structure and relationships between the managed resources and the policies that govern them. They propose a generic and domain independent policy conflict analysis mechanism. A disadvantage of this approach is the naive, pairwise comparison method applied, in order to determine conflicting relationship between policy rules. This increases the time complexity of conflict analysis execution, making it a cumbersome approach for highly conflicting environments such as those we focus on. The first phase of their conflict analysis mechanism, which we see as a time inefficient step, is the manually created information model, which is used to determine the relationships between the potentially conflicting, newcomers and the existing policies. In order to determine a conflicting relationship between two policies, it involves the transformation of the information described in UML model into matrices; following then a number of conflict detection patterns it performs comparisons on policy's subjects, targets and actions attributes in order to define a conflicting relationship. The use of matrices is an impediment with limited expressiveness in defining relationships amongst the core attributes of policies. Moreover, the matrices development, demands to excessively check the correlations (e.g., subset, superset relationships) between all the attributes of the policy rules (i.e., subject, target and so on). For instance, in the case where the subject attributes of two policy rules under conflict analysis refer to completely different sets of resources, which implies nonconflicting relationship between them, their algorithm will excessively check relationships between all other policy blocks (i.e., target, action etc.).

Charalambides et al. in [19, 20] unlike us, use event calculus as a logic-based formalism to describe the managed system. Their policy conflict analysis is comprised of two main aspects: a) the definition of appropriate rules for determining potential policy conflicts and b) the effective deployment of analysis processes in the context of the managed environment. In order for their method to execute they need to know the exact structure of the managed system, and they also need to clearly define and classify all the possible policy conflicts that can occur in this particular system in advance. After that, applying a set policy conflict detection rules that take the form of “meta-rules" they determine conflicting policies. In our point of view this is a cumbersome approach when one deals with highly dynamic environments where the managed system may change dynamically and so do the potentially conflicting policies that govern it. Finally, it is a difficult to extend approach unlike ours.

The work in [13] focuses on a specific system management unlike our domain independent approach. Their conflict analysis model focuses on the dynamic detection and resolution of IPSec security policy conflicts. The formalism that is used in their work to represent IPSec policies and detect conflicts between them is a binary decision tree based on propositional logic. The potentially conflicted policies in their approach manage narrowed and well defined systems where the potential policy conflicts are well determined and specified in advance unlike in our environments of interest. Finally, their binary decision approach lacks expressivity making it not a good match for policy conflict analysis by non-technical users operating at the organizations edge.

Authors in [83] deal with policy conflict analysis of policies in charge of managing networking quality of service (QoS). They analyze low-level policies in terms of both, policy representation and domain of application. They deal with policies that govern routers and packet-level network operations and consist of two basic components: a) policy criterion/condition and b) policy action, focusing basically on the second. The policy-based managed environments are predetermined and static unlike ours. They propose a flexible mechanism for maintaining a conflict free PBMS that governs several parameters along the network path, capable of detecting conflicts in terms of action policy block using fuzzy measurements.

Policy ratification work of [5] proposes a generic and domain independent policy conflict analysis model similar to ours, focusing on the policy condition attribute by proposing algorithms for defining inconsistency, coverage and conflicting relationships between policies given the conditions. However, the main issue of policy conflict analysis that they focus on, is the conflict resolution, which is not our main point of interest herein. They develop policy conflict resolution algorithms, aiming to automate the process based on a mechanism that associates relative priority values to policies, in order to decide which of the conflicting policies to dispose or not.

The conflict analysis model presented in [28] and [30] proposes a generic and domain independent approach similar to ours. The main difference to our approach, is that we propose a discrete and sequential execution conflict analysis model that breaks down the conflict analysis process into discrete steps. When in any of those steps a non conflicting relationship between two policies is determined, the analysis terminates avoiding unnecessary comparisons. Finally, the use of matrices and tree representations that they propose makes it difficult for human policy authors (especially for those with lower technical background), to track the reasons of policy conflicts, whereas in our approach the analysis pushes those reasons in a near to natural language representation to the user making the whole process more transparent.

4.3 Conflict Analysis of High-level Policies

4.3.1 CE & Policy Conflict Analysis

The policy conflict analysis model proposed in this thesis, is applied on policies expressed in CE policy language [70], as presented in detail in Chapter 3. However, for ease of reference, we will summarize the main characteristics of the CE policy language. It is an ontological approach that uses a Controlled Natural Language for defining a policy representation that is human-friendly (CNL representation) and unambiguous for computers enabling policy rules evaluation utilizing the CE reasoner [66]. Moreover, CE is used to define domain models that describe the system to be managed, while those domain model's components are the building blocks of the attribute-based CE policy rules. Apart from the technical tools that the CE provides for developing a reliable and efficient policy conflict analysis mechanism, it also eases policy makers with limited technical expertise, to cope in a more transparent way with complexities associated with policy conflict analysis.

4.3.2 Policy Conflicts Classification

Prior to presenting our approach, we classify the types of generic policy conflicts that can occur in the context of a PBMS based on different parameters; for instance the types of policies, the time frame in which conflicting relationships can be detected, and the correlation between policy conflicts and the domain where policies are enforced.

At first, conflicting relationships between policies can either occur between: a) authorization policies, b) obligation policies, or c) on policies of different types, namely authorization and obligation policies. The second major policy conflict classification considers the time frame in which conflicts can be detected. In that occasion policy conflict detection is distinguished in: a) static (or off-line) and b) dynamic (or on-line). The first are conflicts that can be determined without requiring the policy's execution.

These are conflicting relationships that can be inferred just by analyzing the policies' attributes, regardless the environment that they are applied on. Unlike the static, the dynamic conflicts can only be detected at the policy's execution time, considering also extrinsic parameters such as the state of the managed system or conditions that describe general truth constraints such as time and space. Take for instance two policies P_1 and P_2 where the P_1 permits the policy's subject to access a particular set of targets and P_2 forbids it for the same subject – target set. Having two policies that force opposite actions on the same set of attributes, implies conflicting relation as stated before. Nevertheless, if P_1 has the following condition P_C1: *if Target* T_1 *isFree* then the two policies P_1 and P_2 are only in conflict if the set of target T_1 resource is not used by any other subject.

In addition to the two major classification categories, policy conflicts can also be classified into: *intra-policy* conflicts, which occur due to rule misconfiguration within a single policy, or as it is described as an ill-defined policy and *inter-policy* conflicts; conflicts that occur between multiple applied policies leading to conflicting actions [41]. Finally, policy conflicts can also be distinguished in domain-independent and domain-dependent. The domain-independent refers to policy inconsistencies that occur due to generically opposite relationships between policies' actions. On the contrary, the domain-dependent conflicts group, contains those inconsistencies that are related to the context of the application domain. Consider for instance the following brief scenario inspired from the banking sector where financial transactions are managed through policies. Assume two policies that manage payment requests and payment approval. If the same bank employee (i.e., policy subject) can perform both of those two actions then there is a conflicting relationship between the two policies. The conflict is tied to the application domain and hidden on the policies' action attribute semantics.

4.3.3 Overview of conflict analysis approach - Strengths and Weaknesses

We briefly present here an overview of the CE-based policy conflict analysis model, highlighting its strengths and weaknesses. We then present more in detail the algorithms, their time complexity analysis and evaluation in the following sections.

The goal of the conflict analysis model that we propose, is to determine pairs of conflicting policies that reside in PBMS' policy repository. This requires the application of analysis process between all the possible pairs of policies. Utilizing CE language for expressing policies from both major policy types (i.e., authorization and obligation), the CE-based policy conflict analysis model needs to deal with conflicts that occur in all three different combinations mentioned in Section 4.3.2 (i.e., conflicts between authentication, obligation, or both policy types). As far as the conflict detection time is concerned, the CE-based analysis model deals with both static and dynamic conflicts.

Even though CE allows for defining structures with multiple action causes, in CE policy representation each policy rule has a single action attribute thus, the existence of intra-policy conflicts is impossible to occur. Hence, the conflict analysis model copes only with inter-policy conflicts. Finally, our approach is generic and discipline independent, aspiring to be applied in generic PBMS middleware. As such, it copes with the detection of abstract, domain-independent conflicts. However, this does not mean that it cannot deal with domain-dependent conflicts. This can be achieved by adding a library that specifies semantically controversial policy actions, as explained later. Nevertheless, an in-depth analysis of such extensions is left for future work.

A useful tool from utilizing CE for the development of a policy conflict analysis mechanism is the ability to develop such a tool using a single representation. To the best of our knowledge, unlike any proposed policy conflict analysis approach, the CE-based approach does not require the transformation of the managed system, the formalized representation and/or the policy representation into any other form in order for the policy rules to be analyzed for conflicting relationships. This is because CE provides a representation that can be: a) understood by human user, and so used by policy authors, domain model developers, while b) being amendable for analysis by machines through CE reasoner. Thus, in the CE conflict analysis model:

- The domain model is expressed in CE.
- The policies are expressed in CE.
- The conflict analysis can be performed on these very constructions.

As we are not going to evaluate the real-time execution performance of the conflict analysis tool herein – this is going to be part of future work – we claim that by utilizing the single representation capability of CE-based approach, we reduce the number of steps that are needed for the implementation of an already complex mechanism and potentially the overall time complexity of the analysis process.

Finally, dealing with collaborating and dynamic environments and as a consequence a highly conflicting PBMS, the conflict analysis tool that we propose needs to be time-efficient. Time-efficiency is achieved by the semantics provided by CE and CE policy language. The structure of CE policy language and its seamless relation to the CE formalism that describes the managed system are both exploited for the implementation of a discrete and sequential conflict analysis process. The three steps of the conflict analysis start, by executing first the least time costly process (i.e., the analysis related to policies' subject and target attributes); then follows the analysis regarding the action attribute and the process terminates with the most costly step, this of the optional condition(s) attributes analysis. Part of the final conflict analysis step utilizes the CE reasoner. CE being a research stage project, we believe that there is still room for improvement as far as its reasoning capabilities are concerned. For that reason we tried to decrease the role the slower reasoning execution plays in our conflict analysis implementation.

4.4 CE-based Policy Conflict Analysis

4.4.1 Policy Conflict Analysis Architecture

The CE-based policy conflict analysis implementation resides in two different domains: a) the CE Store processing environment and b) the Conflict analysis module and it is comprised of three algorithmic steps. We call the first and third algorithmic steps “hybrid", due to their dual domain execution nature. The CE Store as it was explained before is used to formally model the policy-based managed system and evaluates the policies on it. In the conflict analysis process it is also used for the implementation of the first stage of the first algorithmic step and the final stage of the third algorithmic step as shown in Figure 4.1.

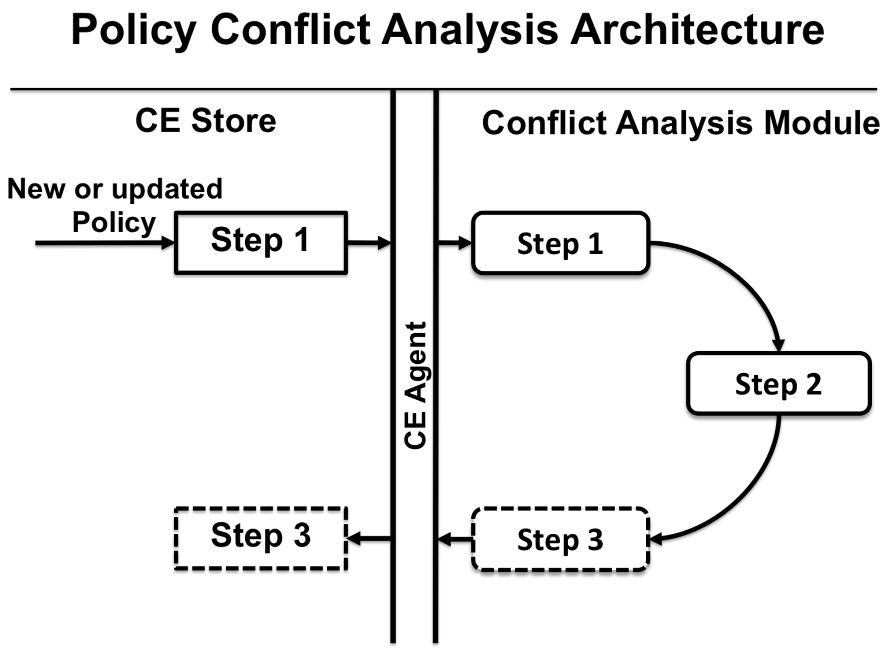


Figure 4.1: Policy Conflict Analysis Architecture

The dotted lines of step 3 indicate its optional execution. The CE Agent plays an intermediate role transferring information (e.g. domain model information) for further computational processing between the two domains. Finally, the conflict analysis module is in charge of the completion of the main computational load of the conflict analysis algorithms as explained below. It gets input from preprocessing applied in CE Store environment and its output declares whether further computation is needed for policy conflicts determination in the CE environment.

The entities of the CE domain model are the potential Subject, Target and Action attributes of a CE policy rule. The resources of the managed system as formatted in the domain model are the potential policies' subject and target grammatical blocks. Given the system's operational flow and capabilities, policies' actions are defined by the relationships between the concepts as described on the domain model. Finally, the domain model's entities are points of reference for policies' conditions development. For example, in an authorization policy rule case, a concept might represent the subject attribute of a policy that is a set of human users with positive authorization rights as action attribute on a set of machinery devices as target attribute. In an obligation policy rule on the other hand, the subject attribute of a policy might be a service entity that has a triggering action on a different service that represents the target policy block. It is worth mentioning here that the high-level policies that we want to apply in our policy conflict analysis are in charge of governing actions of both hard (software and hardware entities) and soft (human) system resources.

A capability of CE, which is a useful tool for policy conflict analysis' implementation, is its ability to build hierarchies of concepts, supporting multiple inheritance of them. As explained in Chapter 3 the creation of a CE domain model is achieved through conceptualization CE sentences, which are interpreted by the CE Store environment. Using the "*is a*" syntax in those sentences one can define concepts, which are specializations and inherit properties of other more abstract ones. For instance, the concepts *Male User* and *Female User* can be specializations of the hierarchically higher concept of *User*.

The major computation workload of the first, hybrid algorithmic step of conflict analysis is executed in conflict analysis module. It receives its input through *Concept*

Selection process applied in CE Store environment. Concept selection process, passes to the conflict analysis module the higher hierarchical concepts of subject and target blocks of the policy to be analyzed up until their root concept (i.e., the most abstract one). Following the previous example, if the subject of a policy that needs to be analyzed is the *Male User* concept, the concept selection passes both *Male User* and *User* concept to conflict analysis module. This way we manage to cover any of the four relationships between subject and target blocks of policies under conflict analysis:

- Equality
- Superset
- Subset
- Correlation or Intersection

The prerequisites for conflicting relationship between two polices are explained in detail below. Briefly, two policies with opposite actions attributes are in conflict if any of those relationships exist between their subject and target sets. Dealing with the more abstract concepts we manage to cover all four of them.

Apart from the four policy grammatical blocks, i.e., the binary Conditions, Subject, Action and Target, each policy has a unique identifier parameter ID and a binary policy type identifier that determines whether it is an authorization or obligation policy. This hextuple representation of policies provides the six necessary components that are used as subject of the policy conflict analysis. Before we proceed with the description of the three steps of the sequential and discrete policy conflict analysis algorithm, we first define the conditions that need to occur in order for two policies to be in a conflicting relationship. The following definitions are based on *conflict of modalities* policies inconsistency type as described in [58, 63], and drive the discrete steps of the policy conflict detection algorithm presented in next Subsection.

Assume two policy rules P_1 and P_2 and their grammatical blocks where P_{C1}, P_{S1}, P_{A1}, P_{T1}, P_{I1}, P_{P1} are the condition, subject, action, target, ID and type attributes of P_1, and P_{C2}, P_{S2}, P_{A2}, P_{T2}, P_{I2}, P_{P2} the respective attributes of P_2.

Definition 1 - 1st Obligatory requirement: *Two policy rules are in conflicting relationship if the intersection between their subject attribute sets and the intersection between their target attribute sets are not empty.* Assuming the two policy rules above P_1 and P_2 that is: $P_{S1} \cap P_{S2} \wedge P_{T1} \cap P_{T2} \neq \emptyset$.

Proof: It is intuitively understood that if the subjects P_{S1} and P_{S2} and similarly for the target attributes of two policies are not intersected sets, that means that the policies refer to different objects. Thus, regardless the relationship between actions and conditions attributes, the policies P_1 and P_2 cannot contradict with each other.

Definition 2 - 2nd Obligatory requirement: *If Definition 1 is satisfied, then two policy rules are in a conflicting relationship if their action attributes have a contradicting behavior.* As in P_1 and P_2 case it must be $P_{A1} \neq P_{A2}$.

Proof: By contradiction, two policies that enforce the same actions P_{A1} = P_{A2} cannot be conflicting as they guide the system they govern towards the same behavior. These kind of policies have a possible redundancy relationship instead, but not a conflicting one.

Note: The inequality sign ($\neq$) between P_{A1} and P_{A2} in Definition 2 declares opposition between actions. The opposition has different meaning depending which of the three types of conflicting policies we deal with. Thus, in conflict detection between binary authorization policies case, the inequality sign has the literal meaning of not equal, where one of the actions has positive authorization sign and the other negative. Thus, it is P_{A1} = $\neg P_{A2}$.

In conflict detection case between authorization and obligation policies, regardless the action of the obligation policy rule, the inequality sign ($\neq$) considers an authorization policy with negative sign ($auth-, notAccess$). It is clear that there is conflicting behavior on a system if one policy obliges an action while the other forbids the access for intersected sets of subject and target sets.

Finally, in the case of obligatory policies the inequality sign requires a more complex analysis because the opposition is expressed through the actions' semantics. In Section 4.3.2 where we classify the policy conflicts, we referred to domain-dependent conflict classification which deals with policy inconsistencies related to the context of the application domain. We illustrated it with a little scenario of a bank employee who given two policies can make a payment request, while he is the same person that approves it. The inconsistency here is between the *make payment* and *approve payment* actions. This kind of actions' conflicts can be dealt with the utilization of a knowledge-base library that statically defines semantic inconsistencies between actions.

Definition 3 - 3rd Obligatory requirement: *If Definitions 1 & 2 are satisfied then two policies are in a conflicting relationship if all of their condition(s) are simultaneously true without causing empty intersections between their subject attribute sets or empty intersection between their target attribute sets.* As in P_1 and P_2 case, if predicates $P_{C1} \wedge P_{C2}$ are simultaneously true then in order for a conflicting relationship to occur must be $P_{S1} \cap P_{S2} \vee P_{T1} \cap P_{T2} \neq \emptyset$.

Proof: A policy allows or forces a set of subjects to perform the declared action on a set of targets if and only if its boolean condition(s) is/(are) true. Otherwise its action is not performed. For two policies that satisfy the requirements of Definitions 1 and 2 and their actions cannot be performed (i.e., have their conditions true) simultaneously on the managed system, they are not in a conflicting relationship. They are in a potentially conflicting relationship. Using predicate logic a formal representation of the proof is as follows. If $P_{C1} \wedge P_{C2}$ is predicate A, an enforced policy is predicate B and a conflicting relationship between P_1 and P_2 is the predicate C then from Definition 3 we have if A $\Rightarrow$ C. The predicates B and A are in a B $\iff$ A relationship. So a negated B implies a negated C namely $\neg$B $\Rightarrow$ $\neg$C.

Note: The analysis of condition(s) policy block is a complicated task. There are two major conditions that can be expressed in CE policy language, the abstract ones that refer to general facts (e.g. conditions related to time) and domain specific conditions that describe constraints regarding the domain model's concepts and attributes (e.g. the age attribute of a user concept). As mentioned before there are policy conflicts that can be detected statically and others that can be only detected at policies evaluation time, which we refer to as dynamic conflicts. To start with, if two policies under analysis satisfy the requirements of Definitions 1& 2 and have no condition attributes (conditions are optional) then they are in a conflicting relationship. If two policies under analysis have one or more conditions then we need to apply the following steps in a pairwise manner.

If the conditions of two potentially conflicting policies deal with the same attributes the computation for defining conflicting cases is executed at the conflict analysis module and the type of conflict detection is still static. For instance, if for two policies P_1 and P_2 their condition attributes P_{C1} and P_{C2} deal with the age attribute of a user concept (i.e., a domain specific condition) or a time related parameters (i.e., abstract condition) and there is intersection between the conditions P_{C1} : 20 < age < 40 and P_{C2} : 30 < age < 40 the P_1 and P_2 are in a conflicting relationship. An intersection in conditions' values implies intersection between subject and target policy block sets as well. Thus, the prerequisites for conflicting relationship as stated in Definition 3 are met.

Finally, if the policies' conditions deal with different conceptual model entities or if only one of the two policies is unconditioned then they need to be evaluated in the CE store, using the CE reasoner. In this case we deal with a dynamic conflict definition and the two policies need to be evaluated on the domain model and have their subject and target sets checked for overlapping. If the intersection between their subject sets or the intersection between their target sets is not empty then there are conflicting relationships between them.

4.4.2 Algorithm & Analysis

Dealing with conflict analysis in collaborative and highly dynamic environments, where the likelihood of conflicting policies is generally high, requires for an efficient mechanism. Moreover, remember that the environments that we focus on are populated by non IT expert users. To cope with these requirements we need to exploit the CNL capabilities provided by CE. First, its ontological nature that embeds knowledge regarding the managed system (i.e., concepts, their attributes and relationships between them) and the policies (i.e., policy grammatical blocks: subject, target etc.) as well as the reasoning capabilities of CE and its user friendly representation. The conflict analysis algorithm is executed in three discrete and sequential steps, where each one of them copes with the three conflict definitions respectively, in defining conflicting relationships between policies.

First step: The first, hybrid step of the algorithm exploits both the ontological nature of CE and its ability to build hierarchies of concepts and the computational flexibility provided by the conflict analysis module. It takes as input a newly created or updated policy's subject and target attributes P_{NewS} & P_{NewT} and as output places the policy P_{New} appropriately in the *Cached policy repository table*. As it is presented in Algorithm 4.1, if needed (i.e., if subject and target entities are not root concepts) it gets also the hierarchically higher subject and target attributes, up until the root concept, executing then the same process. We call this process concept selection. The index integer that defines the appropriate placement of P_{New} in the cached repository table is created by the non-cryptographic, 32-bit, Java hash function. The choice of a cached repository table (i.e., an array of lists), as data structure for improving the efficiency of conflict analysis' execution time, is based on the fact that we deal with the mapping of an arbitrary size of policies to a fixed size table.

The fixed size of the table $i \in \mathbb{N}$ represents the number of potential subject-target sets where $i = N^2$ where N is the number of concepts in the conceptual model. The number of potentially conflicting polices that share the same subject-target attribute set

Algorithm 4.1: Policy conflict analysis - First step

```
1: get($P_{NewS}$)
2: get($P_{NewT}$)
3: addCachedRepository(hashFunction($P_{NewS}$, $P_{NewT}$), $P_{New}$)
4: if $P_{NewS}$ != mostAbstract(S) then
5:     S' = getMoreAbstract(S)
6:     addCachedRepository(hashFunction($P_{NewS'}$, $P_{NewT}$), $P_{New}$)
7: if $P_{NewT}$ != mostAbstract(T) then
8:     T' = getMoreAbstract(T)
9:     addCachedRepository(hashFunction($P_{NewS}$, $P_{NewT'}$), $P_{New}$)
```

is arbitrary as it is dynamically determined by several environmental factors. The hash function takes as input key the string values of the policy's subject and target attributes and returns a unique hash value integer $i \in \mathbb{N}$ where $0 < i < N^2$, that represents the index of cached policy repository table. Thus, we succeed to group together in one list those policies that have potentially conflicting relationships, satisfying the requirement of Definition 1. In more detail, in Steps 1 - 3 the algorithm places appropriately the P_{New} into the cached policy repository table considering the actual subject and target set. Steps 4 - 6 consider the process of policy's placement into cached policy repository table performing the concept selection method for the policy's subject block, while Steps 7 - 9 present the same process for policy's target block.

Second step: The second step of the algorithm is executed in its entirety in the conflict analysis module and deals with Definition's 2 prerequisites. It compares the action attributes $P_{New}A$ of the new/updated policy with the action attributes of those in the same list of cached policy repository table $P_{Old}A$ as shown in Algorithm 4.2. It takes as input the action attribute of the policy under analysis P_{New} and its binary type identifier $P_{New}P$, (i.e., authorization or obligation type).

The output of the second algorithmic step is the definition of pairs of policies with

Algorithm 4.2: Policy conflict analysis - Second step

```
1:  for cachedRepository[index].size() do
2:    if $P_{NewP}$ = authorization then
3:      if $P_{OldP}$ = authorization then
4:        if $P_{NewA}$ != $P_{OldA}$ then
5:          potentialyConflicting($P_{New}$, $P_{Old}$)
6:      else
7:        if $P_{NewA}$ = negativeAuthorization then
8:          potentialyConflicting($P_{New}$, $P_{Old}$)
9:    else
10:     if $P_{OldP}$ = authorization then
11:       if $P_{OldA}$ = negativeAuthorization then
12:         potentialyConflicting($P_{New}$, $P_{Old}$)
13:     else
14:       if actionCoverage($P_{OldA}$, $P_{OldA}$) then
15:         potentialyConflicting($P_{New}$, $P_{Old}$)
```

contradicting actions. As explained in Definition's 2 note section, the algorithm first specifies which of the three cases of conflicting policies it deals with (i.e., authentication – authentication, authentication – obligation and obligation – obligation) and then infers the relationship between their actions. The first case (authentication – authentication), compares the action strings. The second case (authentication – obligation) deals with the authorization policy type and checks whether it has a negative sign and finally in the third case (obligation – obligation) it applies the action coverage method utilizing the conflicting action library. The Steps 2 - 4 presents the algorithmic sequence for the first policy conflict case (i.e., authentication – authentication) and defines the actions' block relationships in Step 5. Steps 7 - 8 deal with the second case (authentication – obligation) when P_{NewA} has a negative sign, while Steps 10 - 12 deal with the same conflict detection case, although this time the P_{OldA} is under investigation for

Algorithm 4.3: Policy conflict analysis - Third step

```
1:  if $P_{New}$ hasCondition = true then
2:      for all $P_{NewC}$ do
3:          if $P_{Old}$ hasCondition = false then
4:              dynamicConflictAnalysis($P_{NewC}$, $P_{OldC}$)
5:          else
6:              for all $P_{OldC}$ do
7:                  if $P_{NewC}$ == $P_{OldC}$ then
8:                      staticConflictAnalysis($P_{NewC}$, $P_{OldC}$)
9:                  else
10:                     dynamicConflictAnalysis($P_{NewC}$, $P_{OldC}$)
11: else
12:     if $P_{OldC}$ hasCondition = false then
13:         inConflict($P_{NewC}$, $P_{OldC}$)
14:     else
15:         for all $P_{OldC}$ do
16:             dynamicConflictAnalysis($P_{NewC}$, $P_{OldC}$)
```

negative authorization sign. Finally, Steps 14 - 15 deal with the third case which infers conflicting relationship between P_{New} & P_{Old} if the action coverage process defines so.

Third step: The third step of the algorithm computes Definition 3 conditions and is executed in the conflict analysis module (in the case of static conflict detection case) or both the conflict analysis module and CE Store processing environment (in the case of dynamic conflict detection case). It takes as input the output of Algorithm 4.2, (i.e., the pairs of potentially conflicting policies P_{NewC}, P_{OldC}) and follows the steps described in Definition's 3 note section as presented in Algorithm 4.3. In more detail Steps 2 - 4 deal with the case of dynamic conflict analysis when the P_{OldC} is empty (i.e., no policy condition). In Steps 6 - 8 it is presented the static policy conflict analysis when

P_{NewC} and P_{OldC} refer to the same domain model concepts while in Steps 9 - 10 the dynamic conflict analysis is executed when both P_{NewC} and P_{OldC} are not empty and refer to different concepts. Finally, Steps 12 - 13 define conflicting policy relationship when both P_{NewC} and P_{OldC} are empty and Steps 15 - 16 present the dynamic conflict detection in the case where P_{NewC} is empty.

4.5 CE Policy Conflict Analysis Evaluation

The validity of the CE-based policy conflict analysis mechanism is supported by the theoretical analysis in the previous section. A demo video [71] of policy conflict detection prototype utilizing the web interface of CE Store can be found at `https://users.cs.cf.ac.uk/C.Parizas/Demo.swf`. To test the performance of the CE-based policy conflict analysis mechanism we devised a set of experiments in order to compare the number of total comparisons needed for it to determine conflicting policies compared to the number of total comparisons needed for a naive pairwise conflict analysis to achieve the same goal.

We compare our approach to the naive pairwise model because the vast majority of the proposed conflict analysis tools follows it. To the best of our knowledge only [29] proposes an efficient analysis, i.e., not a pairwise approach, in generic and domain-independent conflict analysis tool as a continuation to the work in [28]. Given the heterogeneity between ours and prior work as presented in Section 4.2 in terms of implementation, tools' utilization and policy representation, it is hard to choose a prior and ideally the most efficient approach and compare ours to.

Our evaluation focuses on measuring the performance of our approach when system parameters such as the number of concepts involved in PBMS as well as the number of policies scales up. We assume that a PBMS in charge of managing a system with fewer concepts (i.e., fewer entities to be managed) will contain fewer policies as well, compared to a PBMS that manages system with more concepts and thus policies. Before we start to explain the experimental work and its results, we first present the analysis of our algorithm comparing it with the naive pairwise approach analysis, focusing on their time complexity.

Assume a PBMS that is comprised of n policies where $n \in \mathbb{N}$. In order for the naive pairwise algorithms to compare all n policies with each other and given that the analysis is applied on SAT policy syntax, each policy needs to make four types of comparisons (i.e., condition, subject, action and target), with all the existing policies in PBMS. Given that by comparing policy rule P1 to P2 is the same comparison as if comparing P2 to P1 we consider a decent pairwise opponent, which needs time T given by the Equation 4.1

$$T(n) = 4 \times \frac{n(n-1)}{2} \tag{4.1}$$

namely a quadratic execution time where time complexity is O(n^2). The approach we propose for a list of n policies where $n \in \mathbb{N}$ has three discrete and sequential steps of execution. The first step utilizing the cached repository table data structure derivative of the hash function. in just 1 step, it implements the comparison of subject and target blocks by placing policies that share the same sets of those attributes in the same list ready to be compared in terms of the rest two policy blocks (action and condition(s)). Given that the comparison for the rest two policy objects is a pairwise process as well, the overall time T for the proposed algorithm execution is given by the Equation 4.2.

$$T(n, N, m, k) = 1 + \sum_{i=1}^{N^2} \frac{m_i(m_i - 1)}{2} + k \tag{4.2}$$

namely a quadratic execution time where time complexity is O(n^2) similar to pairwise approach. The variable $N \in \mathbb{N}$ is the number of concepts that comprises the conceptual model, variable m_i represent the i$^{\text{th}}$ set of m policies that share the same set of subject and target policy blocks and finally variable k represents the number of pairs of policies that satisfy Definitions' 1 & 2 prerequisites. However, the variables m, $k \in \mathbb{N}$ have the following relation compared to n: $n > m > k$. This is a result of the discrete and sequential execution of our algorithm. The naive, pairwise approaches compare all the policy blocks with each other while our approach gradually, as it evolves excludes policy comparisons considering only policies that are potentially conflicting given the prior and current algorithmic steps.

For the evaluation of CE-based policy conflict analysis mechanism's performance, we run a set of experiments simulating its application on three different PBMS scenarios considering different values for two parameters. In the first case we assume the management of a rather simple system which is comprised of 5 different concepts that interact with each other through 4 different types of relationships. Therefore, the policy subject and target blocks can be any of these 5 concepts while the possible policy action can be any of the 4 different relations.

Moreover, in the first experimental case we assume a number $n = 100$ of policies in charge of managing the aforementioned system. All the policies are randomly created considering the sets of possible condition, subject, action and target attributes of the managed system. In all three scenarios the policies have only one condition and all conditions' analysis is performed in conflict analysis module (i.e., static conflict evaluation) for simplicity.

In the second case we experiment assuming a managed system composed of 10 different concepts and same number of relationships while the number of polices in charge of governing the system is now $n = 1000$. Finally, in the last set of experiment we assume a system composed of 20 concepts and same number of relationships, while the number of policies is $n = 10000$. We repeat each one of the three simulated cases 10 times averaging the measurements and rounding the decimals.

We present the performance evaluation results in Figures 4.2, 4.3 & 4.2. All three of them compare the number of comparisons needed for each one of the policy rule blocks (i.e., subject, target, action, condition) as well as the total number of comparisons needed in order for the CE-based and the naive pairwise approaches to determine

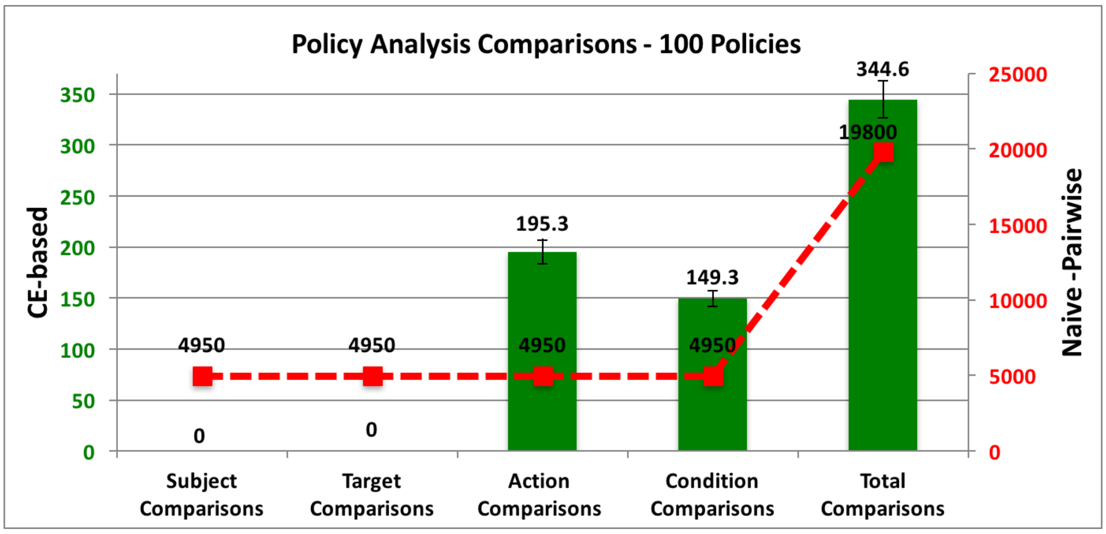


Figure 4.2: Policy Analysis Comparisons - 100 Policies

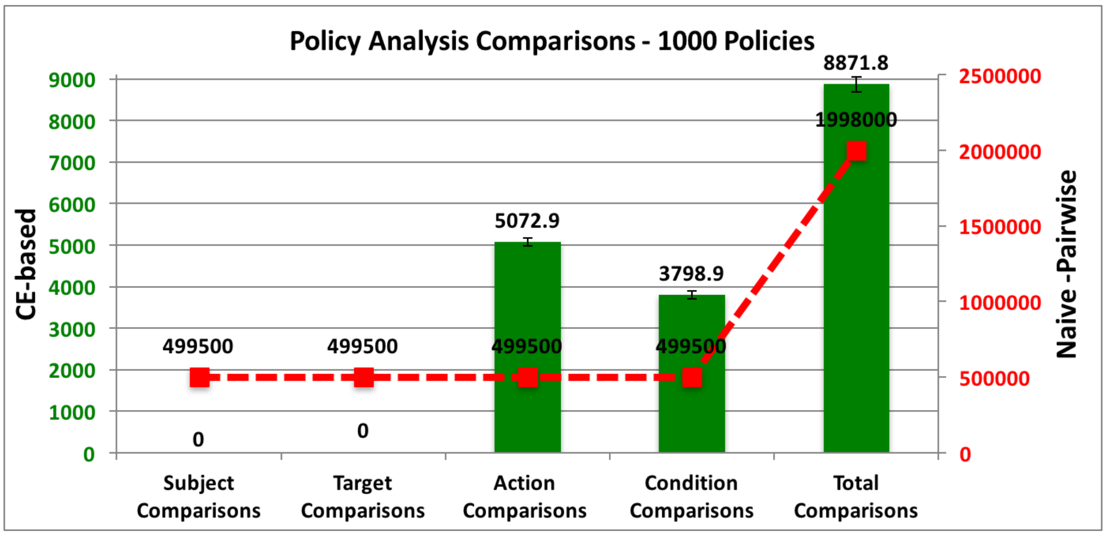


Figure 4.3: Policy Analysis Comparisons - 1.000 Policies

conflicting policy relationships. Due to the different scale between the CE-based and the naive pairwise approach, all three figures present two y axis. The left hand side one, as indicated, presents the CE-based comparisons and is paired with the clustered columns chart, while the right hand side refers to the naive pairwise comparisons paired with the line with markers chart. The error bars on clustered column charts represent (+ / -) 1 Standard Deviation.

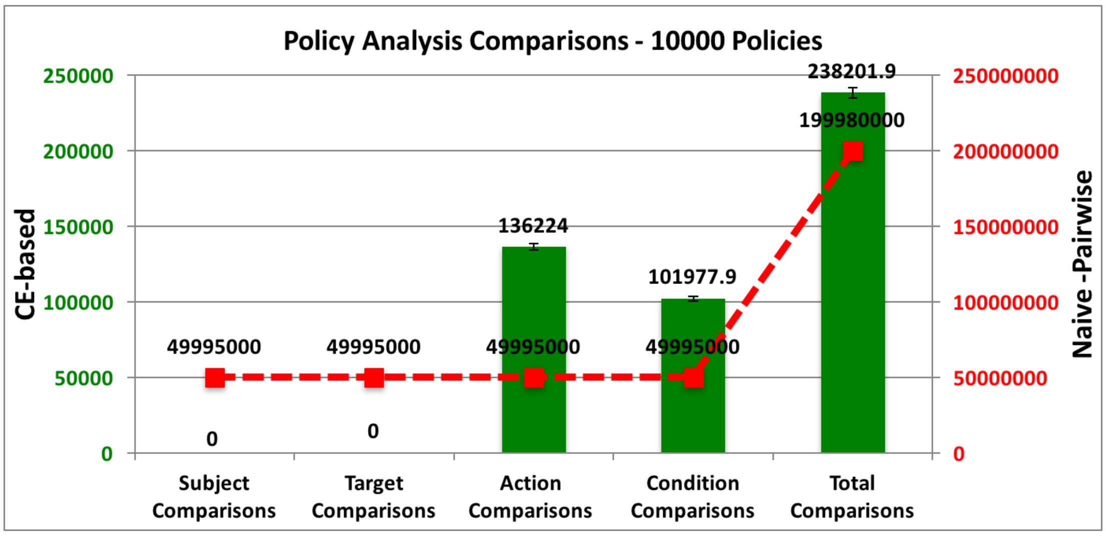


Figure 4.4: Policy Analysis Comparisons - 10.000 Policies

Figure 4.2 presents the results for the first experimental case; Figure 4.3 and 4.4 respectively present results for the second and the third experimental case as described above. The results for the pairwise approach evaluation are products of Equation 4.1 and they are a fixed numbers, given the managed system parameters, such as policy attributes and number of policies in charge, so they lack error bars. At a glance in all three cases the CE-based approach outperforms the pairwise approach by a large margin. See that the CE approach needs almost no time to execute the comparisons for the first two policy blocks, which actually means that our approach needs almost no time (one computational step) to make half of the overall comparisons, given that there are in total 4 different policy rule blocks that need to be processed. Thus, the CE-based approach reduces the number of comparisons compared to the pairwise approaches to half.

Having the policies grouped into i number of lists with policies that share similar subject-target pairs we divide the input number m of action attribute comparisons almost by a factor of i (it is not exactly i because the lists are not balanced in terms of number of policies). And finally, in the third stage we proceed to the pairwise comparison for the condition attribute only for those policies that comply with Definitions

Table 4.1: CE-based Vs Pairwise: Total Comparisons Ratio

Policies: 100	Policies: 1000	Policies: 10000
CE-based=Pairwise/57	CE-based=Pairwise/225	CE-based=Pairwise/840

1 and 2 prerequisites. In other words, the proposed approach decreases the number, sparing the system from unnecessary comparisons.

Finally, in Table 4.1 in order to clearly visualize the superiority of CE-based against the pairwise approach we present the ratio of the total number of comparisons needed in both cases, CE-based and pairwise. It turns out that in order for the pairwise approach to define conflicting policies in first experimental case it needs to compute 57 times the comparisons the CE-based approach need. As the number of policies and domain model concepts increase the ratio decreases as well. Thus, in the second case the pairwise needs 225 times the comparisons the CE-based approach needs, while in the most extreme case the difference between pairwise and CE-based approach goes up to 840 times more comparisons.

4.6 Discussion

We conclude that our approach scales better as the number of entities to be managed increases compared to the naive pairwise approaches. This is a trend in contemporary IT systems development today. The rapid evolution of the Internet and the increasing complexities and heterogeneity of modern networking technologies increase the number of resources to be managed posing a significant challenge to systems' management models. Hence, the proposed policy conflict analysis approach seems to be a viable solution to coping with scaled systems' management.

Once a pair of conflicting policies is being defined, the conflict then needs to be resolved, either by refining the conflicting rules in a way that they do not contradict with each other or by rejecting one of them and keeping the other, which is the model that is widely applied to date. The model which is used in most of the conflict resolution cases is inspired by legal theory and practice applying some minor adjustments following the three doctrines as described in [98]. These doctrines are shortly described as follows:

- Lex superior: it gives priority to the most important policy, that can be for instance a policy issued by a super user compared to one with lower credentials, or the negative authorization policy against a positive one there are numerous modifications.
- Lex posterior: it gives priority to newer policy over an older one.
- Lex specialis: it gives priority to a more specific policy over a more general one, that can be a policy with more conditions for instance.

There are few exceptions to that such as the work in [6], which proposes a policy conflict resolution algorithms, aiming to automate the process based on a mechanism that associates relative priority values to policies.

The policy conflict resolution is outside the scope of this thesis. The way we proceed with the task of resolution is to push the pairs of conflicting policies to the policy makers (hence the semiautomatic approach) and let them decide which one to dispose and which to keep. And this is a transparent way, utilizing the user friendly representation of CE and the rationale explanation of policies as presented in Section 3.6.

We adhere to this approach simply because: a) the scenarios that we cope with are sensitive (emergency response, military etc.) and b) we believe that a static conflict resolution approach based on those doctrines above is not suitable for highly dynamic environments. Imagine for instance a policy-based managed system involved in a life critical operation that relies on such simple and static rules when the environment is highly dynamic and affected by many factors. That could be catastrophic.

Thus, we opted to keep the human-in-the-loop, even though it makes the conflict resolution, and as a consequence, the overall policy conflict analysis process slower. We believe that the need for more sophisticated solutions is needed in general and this is part of future work.

Chapter 5

Interest-based Negotiation for Policy-based Asset Sharing in Collaborative Environments

5.1 Introduction

Resource sharing is an important but complex problem to be solved. The problem is exacerbated in a coalition context due to policy constraints placed on the resources. Thus, to effectively share resources, members of a coalition need to negotiate on policies and at times refine them to meet the needs of the operating environment. Towards achieving this goal, in this chapter we propose a novel policy negotiation mechanism based on the interest-based negotiation paradigm. Interest-based negotiation promotes collaboration when compared with more traditional negotiation approaches such as position-based negotiations.

5.2 Background & Related Work

The first computer applications for supporting bilateral negotiations were developed in late 1960s [31]. The reason for their emergence was to assist human negotiators to overcome weaknesses related to negotiation process such as cognitive biases, emotional risks, and their inability to manage complex negotiation environments. Although there is rich literature on negotiation protocols in autonomous, Multi-agent Systems (MAS), there is very limited and no mature work done on policy negotiation.

Briefly, an agent in the context of MAS, is perceived as a software computational entity, capable of possessing the properties of autonomy, social ability, reactivity and proactiveness [103]. In order for MAS agents to cooperatively solve problems, a comprehensive interaction is needed. Negotiation is an effective agent interaction mechanism, enabling autonomous bidirectional deliberation in both situations of competition and cooperation. For the development of sophisticated, negotiation models there are three areas that need to be considered: a) the negotiation protocols that define the rules of interaction amongst agents, b) the negotiation objects that contain the range of issues on which agreements must be achieved and c) the negotiation decision making models that guide agents' concession stance [35].

Initially, automated negotiation has received considerable attention in the field of economics, utilizing the analytical methods of game theory [61], aiming to calculate the equilibrium outcome before the negotiation game is played. While interesting conceptually, game theoretic approaches have been criticized for assuming: a) complete and common information and b) perfect and correct information. However, most real world problems are cases of imperfect, erroneous and incomplete information where revelation is not realistic [72].

Heuristic negotiation approaches started to being studied, to cope with the computationally expensive game theoretic ones, which were based on unrealistic assumptions, considering for instance that each agent has unlimited computational resources and time. Thus, they focused on producing good, but efficient negotiation decisions, as opposed to the optimal and inefficient ones provided by predecessors [50]. The two basic limitations of heuristic approaches were: a) the underused agent communication and cognitive capabilities (e.g., agents' rejection as a feedback when a negotiation agreement is not achieved) and b) the statically defined agents positions (i.e., each agent has a clearly defined and static position) [72].

Argumentation-based negotiation (ABN) has been introduced as a means to enhance automated negotiation by exchanging richer information between negotiators. Interest-based negotiation (IBN) is a type of ABN that describes a mechanism where negotiating agents exchange information about the goals that motivate the negotiation action [37].

We see the role of PBMS in managing large, complex and dynamic systems as of a high importance and the existence of sophisticated ways to do so imperative. We believe that the integration of an effective negotiation mechanism on a PBMS works towards this direction. Moreover, no work had previously attempted to bring the IBN paradigm into policy negotiation.

The authors in [86] focus on the requirements of policy languages, which deal with trust negotiation and pay attention to the technical aspects and properties of trust models to effectively negotiate with access requests. They do not research any of the aspects of policy negotiation and the scenarios they deal with are less dynamic compared to our problem domain. Authors in [45] propose an architecture that combines a policy-based management mechanism for evaluating privacy policy rules with a policy negotiation roadmap. The work is very generic and does not provide clear evidence of any effectiveness of the proposed approach, while lacking any evaluation. To the best of our knowledge [22], is the first work that looks into policy negotiation and covers the area in depth. It also looks into collaborating environments and introduces the notion of ABN in policy negotiation. However, it focuses on a very specific application domain in which it deals with writing insurance policies. The whole process is based on a static approach maintaining a common and collaborative knowledge base.

The work discussed in [92] has several similarities to our work; it deals with cooperating environments and a PBMS is employed in support of service composition in a distributed setting. The authors have used a negotiation framework to effectively compose services. Finally, [21] proposes a policy negotiation approach and presents its architecture. It lacks of any evaluation while it does not consider either multi-partner or dynamic environments, following the PBN paradigm.

The work discussed in [92] has many similarities to our work. The main difference with the work proposed herein is that the objective of the negotiation performed in [92] is the services that are managed by policies, not the policies themselves. We believe that in order to decrease the management overhead the objective of negotiation should be the policies. This is because policies are the core of PBMS and the logical component where the system's management resides.

5.3 Interest-based Policy Negotiation Scenario

Below we provide illustrative scenarios to motivate the use of IBN in policy negotiation in resource sharing situations. In Subsection 3.1 we revisit the classic orange–chefs scenario discussed in best-selling book *Getting to YES* [37] and then expand it to an opportunistic, mobile resource sharing scenario in Subsection 3.2.

5.3.1 The Chefs-Orange Scenario

Two chefs that work in the same kitchen, want to use orange for their recipes. Unfortunately, there is only one orange left. Instead of starting negotiating on who is going to get the orange or portion of it (as in a zero-sum, PBN approach), the two chefs opt to follow an IBN inspired approach. So, they ask each other why they need the orange for. In other words, they try to better understand their underlying goals of using the orange. Answering the “why" question it turns out that one chef needs only the orange flesh (to execute a sauce recipe) while the other needs only its peel (for executing a dessert recipe) leading them to share the orange accordingly, achieving a win-win negotiation outcome.

5.3.2 Asset Sharing Policy Negotiation

An individual P2 wants to access a smartphone device SMD owned by an individual P1. However, P1 has a set of restrictions which are captured by policy set R on how to share SMD with other people. These restrictions may reflect privacy concerns (e.g., by accessing their smartphone, one could have access to their photos). For the sake of clarity, in this example, we assume that the set R contains the following policy constraint R1: *do not share the device SMD with anyone else but its owner P1.* When P2 asks for permission to use the physical device SMD, R1 prohibits this action. Ostensibly there is little room for negotiation here, if one follows a PBN approach with the current set of policies.

However, by applying the IBN and trying to understand the underlying interests of the involving parties, we believe the situation could be handled in a satisfactory manner for both parties. For example, asking the “why" question it turns out that P2 needs a data service (as opposed to the physical device) in order to execute the task of *Email submission* and P1 does not mind sharing a data connection as a hotspot with a trusted party; if P1 could get to know why P2 needs the device for, the situation could be solved to the satisfaction of both parties. All an IBN mechanism needs to do in this case is to introduce another policy – actually a refinement of the existing policy – to R1 to say that data service can be shared among trusted parties. We argue that in such cases, by understanding the situation and broadening the space of possible negotiation deals, one can reach a win-win solution.

The intuition behind ABN is that the negotiating parties can improve the way they negotiate by exchanging explicit information about their intentions. This information exchange reveals unknown, non-shared, incomplete, and imprecise information about the underlying attitudes of the parties involved in the negotiation [81]. Think for instance, a negotiation case where two negotiators after exchanging offers are very close to achieve an agreement, but lacking this extra information they give up moments before achieving it.

As stated earlier, IBN is a type of ABN where the negotiating parties exchange information about their negotiation goals, which then guide the negotiation process. Thus, the why part of the intention is of major importance when compared with the what part. We would say, that the IBN is more of a negotiation shortcut method rather than a typical negotiation process. By attacking the problem of negotiation, IBN could potentially skip the proposals making, the options trading and the need for negotiating parties to offer concession as in PBN cases. Instead of trying to negotiate on a fixed pie, it tries to find alternatives so that to expand it. In the next section, we shall introduce our IBN-based policy mechanism and provide our intuition behind the approach.

5.4 Interest-based Policy Negotiation Mechanism

Designing and developing intelligent tools and protocols for enhancing the negotiation process amongst human negotiators, needs to focus on achieving some desirable outcomes that are secured by meeting a set of systematic properties such as: guaranteed success (i.e., negotiation mechanism that guarantees agreement), simplicity (i.e., eases negotiation decision for the participants), or maximizing social welfare (i.e., maximization of the sum of payoffs or utilities of participants) to name a few. A complete list of desirable negotiation outcomes and evaluation criteria as described throughout the literature can be found in [84]. The main objective of the negotiation mechanism proposed herein, is the maximization of social welfare.

In scenarios that often suffer from resource scarcity (i.e., environments where resource demand exceeds supply), and many user tasks may be competing for the same resource in order to be served, like those described in Section 1.4, the formation of coalitions offers alleviation by bringing more resources to the table. The relationships between coalition parties in those scenarios are mostly peer-to-peer (P2P), but we do not assume fully cooperative relationships. Coalition partners often pursue cooperation but they do not want to share sensitive intelligence that can deliver greater value to the opponents

[56]. In literature this kind of relationship model, where parties have cooperative and competitive attitudes from time to time, is called coopetition [17]. The PBMS and its sets of policies is in charge here, playing a regulative role in order to keep balance between asset sharing and asset “protection".

The stricter the partners’ policies are, the higher the barriers towards collaboration are set. This is where the IBN mechanism comes in, trying to lower these barriers in order to establish better collaboration through asset sharing (i.e., increase overall the number of served tasks and thus increase the social welfare) while maintaining the compromise from the asset owners point of view at the same levels.

The mechanism presented herein allows negotiation on policies with minimal human intervention. In traditional system management, policies associated with PBMS are static (or rarely change); these systems, however, fail miserably in dynamic environments where policies need to adapt according to situational changes. We note that it is not prudent to assume human operators in these environments that can effectively be on top of every change to manage PBMS(s) effectively; they require automated assistance.

Summarizing the intention behind applying an IBN mechanism, on policy regulated asset sharing, it considers a cooperative negotiation approach, for strict policies refinement, that aims to: a) maximize social welfare by increasing the overall usability of collaborating assets while b) remaining faithful to existing authorization policies, maintaining their core trends. Utilizing such a tool, a multilateral policy transformation can be achieved establishing a more effective PBMS, considering input and criteria from multi-party formations, for the benefit of the coalition. The product of IBN execution is a new refined authorization policy rule. The IBN mechanism when refining the strict policy, considers the interest of both: resource owner and resource requestor. As for the negotiation’s protocol, each negotiation session considers sets of two negotiators, so we deal with a bilateral negotiation mechanism. The issue that needs to be settled through the negotiation process, is the granting (or not) of access to non-sharable assets. From that perspective the protocol deals with a single-attribute negotiations.

5.4.1 Policies Under Negotiation

The proposed policy negotiation framework is applied on authorization policies expressed in CE policy language [70]. The CE policy language has been presented in detail in Chapter 3. Very briefly, it is used to define domain models that describe the system to be managed. The domain models take the form of concept definitions and comprise objects, their properties, and the relationships among them. These domain model components are the building blocks of an attribute-based policy language expressed in the very same CE representation.

Each policy rule follows the if – condition(s) – action form and consists of four basic grammatical blocks: Subject, Action, Target and boolean Condition(s).

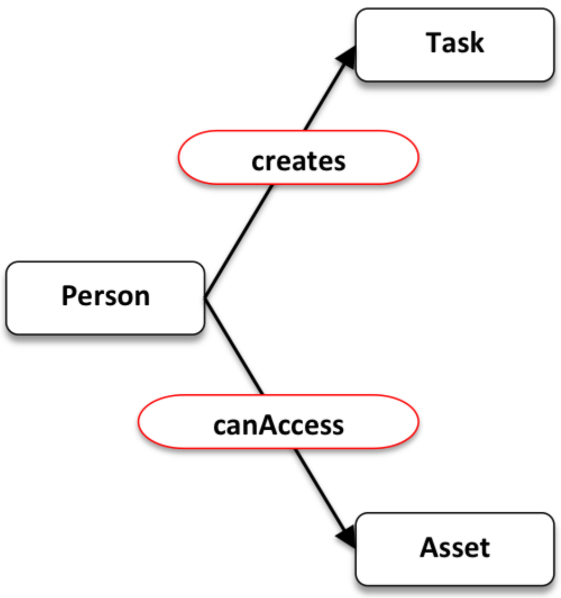


Figure 5.1: Authorization Policy Negotiation Scenario: Domain Model

The utilization of CE here is two-fold. It is not only the user friendly formal representation of a) the system to be managed and b) its policy-based management but it also helps decision makers who lack technical expertise cope in a more transparent way with the complexities associated with policy negotiation. This is by comparing the original to the refined policy in a more user friendly way. Figure 5.1 provides a graphical depiction of the CE-based domain model, which describes the smartphone access scenario of Section 5.3.2 , while the CE representation of policy R1 is shown below.

```
Policy R1
If
( there is an asset A named SMD ) and
( there is a person P named P1 )
then
( the person P canAccess the asset A )
.
```

5.4.2 IBN Integration into Policy Regulated Asset Sharing

The role of policies in managing a system, is to guide its actions towards behaviors that would secure optimal system outcomes. Different users have different rights, relationships and interests in regards to deployed coalition resources. Non-owner users want to gain access to resources in order to increase the probability of serving their tasks' needs, while owners want to protect their resources from unauthorized users. There is therefore a monopolistic resource usage case. The proposed negotiation approach considers both concerns in a single mechanism providing a mechanism that pursues a win-win negotiation outcome for any sets of negotiators. In other words, it tries through negotiation to redefine what is a suboptimal system outcome given: a) the currently-deployed resources, b) the user created tasks' needs and c) the policies themselves.

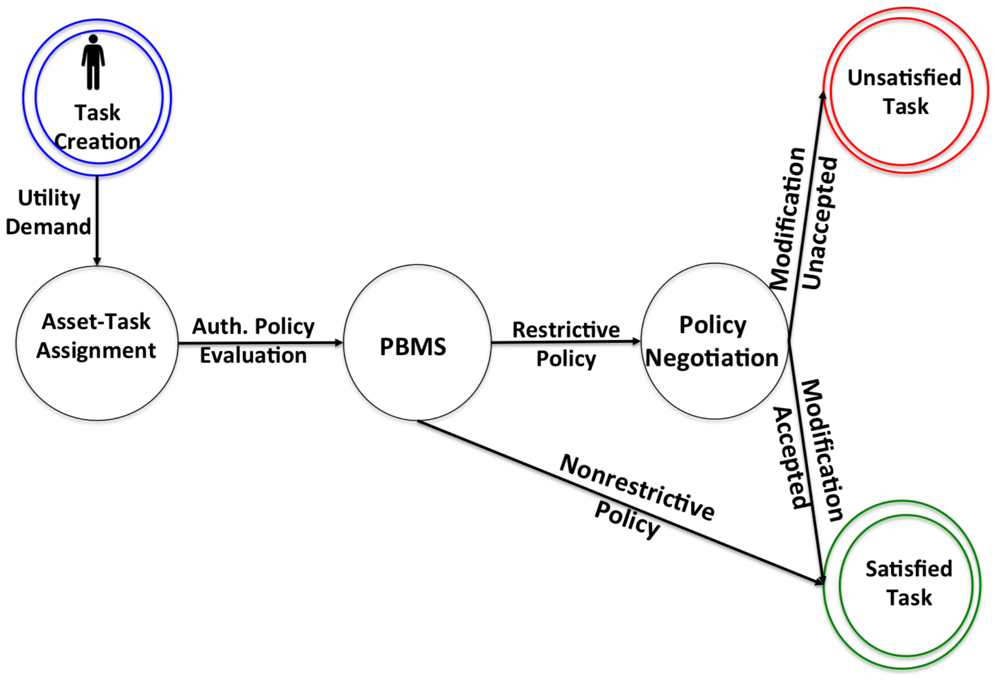


Figure 5.2: Interest-based policy negotiation and task implementation

The finite state diagram of Figure 5.2 provides a depiction of the role the policy negotiation tool plays in the tasks' implementation of collective endeavors. The IBN is an additional step of intelligence compared to how the process flows in Chapter 2. The human task creator, wanting to serve their appetite for information, creates tasks that require a utility demand. The asset-task assignment component is in charge here for optimizing the task utility by allocating the appropriate resources (information-providing assets) to each task. The PBMS component is responsible then for evaluating and enforcing authorization policies made by multi-party collaborators. In the case of a non-restrictive authorization policy the task creator gets their task served. If the policy rule is restrictive, the policy negotiation component takes over. It modifies the policy rule accordingly, and passes it to the asset owner for confirmation. Given the asset owner's decision the task is then either satisfied or unsatisfied.

5.4.3 IBN Enabled PBMS

The policy negotiation framework can be integrated into a PBMS as a plug-in, enabling negotiation in a policy enforcement process. A PBMS, as defined by standards organizations such as IETF and DMTF consists of four basic components as shown in Figure 5.3: a) the policy management tool, b) the policy repository, c) the policy enforcement point, and d) the policy decision point [106]. The policy management tool is the entry point through which policy makers interface (write, update and delete) with policies to be enforced on the system. The policy repository is a specific data store where the policies generated by the management tool are held (step A1). The PEP is the logical component that can take actions on enforcing the policies' decisions, while the PDP is the logical entity that makes policy decisions for itself or for other system elements that request such decisions. Triggered by an event that needs policy's evaluation the PEP contacts PDP (step A2), which is responsible for fetching the necessary policy from policy repository (step A3, A4), evaluates it and decides the actions that need to be enforced on PEP (step A5).

In addition to the four basic PBMS elements, Figure 5.3 also includes a human-in-the-loop element, representing the roles played by the asset requestor and owner in the negotiation process. The additional component where the IBN framework resides is called Policy Negotiation Point (PNP) and lies between the PEP and PDP, interfacing also with the human-in-the-loop element. As mentioned before, the PNP is triggered to attempt to refine authorization policies when a user creates a task that cannot be served due to restrictive policies. The dashed lines show optional communication between the PBMS components, which is only established when a policy negotiation incident occurs. The red numbered part of the figure (flow paths which are prefixed by A's) describe the typical PBMS operational flow, while the green part (flow paths which are prefixed by B's) replace step A5 (red, dotted line) with the policy negotiation extension. Note that the separation between the components can be only logical when they reside in the same physical device. When PNP detects a restrictive policy (step B5) it refines

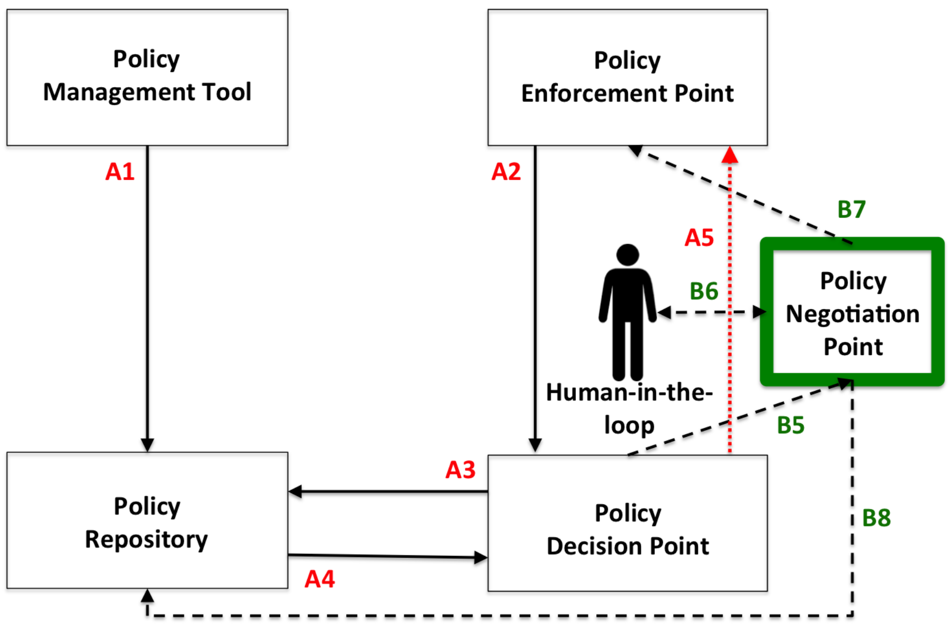


Figure 5.3: IBN extended PBMS

it following the steps described in the following section and passes it to the asset owner for confirmation (step B6). If the asset owner accepts the refined policy rule, it is then pushed to PEP for enforcement (step B7) and either is stored in Policy Repository permanently (step B8) or can be only enforced once and then be discarded. This is on asset owner's jurisdiction. Otherwise step A5 is executed as before.

5.5 Achieving IBN through Policy Refinement

In general, negotiation protocols contain the set of rules that manage the interaction between negotiating parties [46]. These rules define who is permitted to participate in the negotiation process and under what conditions (i.e., negotiating and any non-negotiating third parties). The rules also manage the participants' actions throughout the process.

The negotiating parties in our scenario are essentially decision makers who generally lack negotiation expertise. Thus, the IBN mechanism tries to take, as much as possible, the negotiation weight off their shoulders rather than providing them the means for making proposals and trade options themselves. However, it does not exclude them completely from the negotiation process as in fully automated models. To achieve such behavior it simply applies the IBN principles described in Chefs-Orange scenario of Section 5.3.1, exploiting the domain model's semantics, the semantics of the polices and the seamless relation between them as they both share the same CE representation.

The objective of the negotiation is the restrictive policies themselves. Asking the why question, as in Chefs-Orange scenario, to the asset requestor side, the PNP gets as a reply the reason why they need the asset for (i.e., to get their task served). Asking the why question to the asset owners/policy authors side, it gets the reasons why they do not want to grant access to their assets respectively. The prerequisite for the PNP operation here, is to have full and accurate knowledge of the managed system. This is achieved by having unlimited and unconditional access to both domain model and policy rules of Policy Repository. Unlike the majority of the proposed PBN approaches, the human-in-the-loop negotiators in our case are ignorant of the preferences of their opponents, neither make assumptions about or try to learn them, while their knowledge in terms of the domain model reaches only the ground of their own expertise and ownership.

Utilizing CE as the formal representation for describing the system to be managed, and the representation for expressing authorization policies, eases the human-machine communication (i.e., communication between PNP and non-IT expert negotiators) for exchanging information regarding the negotiation process in a transparent way. The human-machine communication through CE conversational agents has been described in [78]. Briefly, it is a communication protocol, that supports theory of conversations from full natural language to a form of Controlled CNL amenable to machine processing and automated reasoning, including high-level information fusion tasks. It supports three forms of interactions between: 1) human –> machine (human initiates an interaction in NL and the machine feeds back CE, prompting the human to refine the CE and agree an unambiguous CE form of the content), 2) machine –> human (interaction here is to inform a human or ask them for information) and 3) machine –> machine (exchange information between software agents through an exchange of CE content).

However, trying to automate as much as possible the negotiation process, the why question is rather rhetorical here (i.e., PNP does not require input from user). In the asset requestor's case, the answer to the why question is quite simple and straightforward and the PNP is aware of it just taking input from the Asset-Task Assignment component of Figure 5.2. The asset requestor clearly wants to access the asset in order to get their task served. Hence, a desired negotiation outcome as far as the asset requestor is concerned, is the derivation of a refined policy that has them included in the set of *Subject* policy block, with a positive access (i.e., canAccess) *Action*, to a *Target* set that includes the prohibited asset capable of serving their task's needs.

Inferring the answer to the why question from the asset owner's side, for understanding their interests and broadening the negotiation space, is a more challenging task. In general any application of authorization systems, aims to specify access rights to resources. Thus, a simple answer would be including the reasons why asset owners want to decline access rights to their resources in a negative authorization policy, or in contrast, the reasons for granting access to their resources in a positive authorization policy. Thus, the why question from the asset owner/policy maker side can be extracted as the rationale of a policy rule. Combining the definitions from [65, 27] we conclude that rationale is the reasoning pathway from contextual facts, assumptions and decisions, through the reasoning steps, which describes the development of an artifact including details of why it was designed.

Looking carefully at a policy rule, its rationale is basically described from the policy's *Condition(s)* block. The policy R1 of Section 5.4.1 is rather a simple one referring deliberately to a simple scenario and this might not be easily inferred. Considering other more complex policy rules with several conditions describing for instance constraints such as the age of the requestor or their expertise this is easier inferred[1].

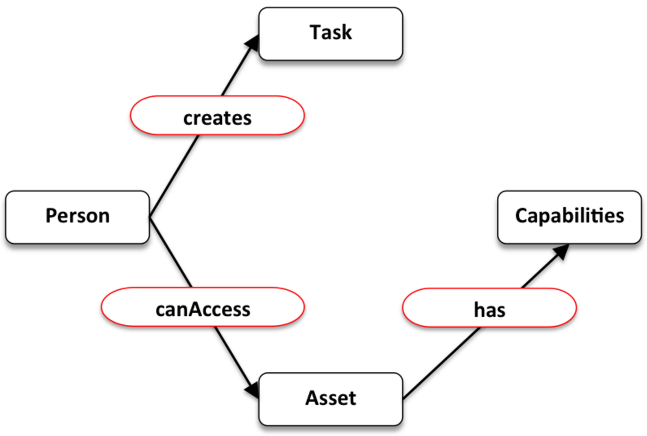


Figure 5.4: Ontology Modification: Step one

However, this is not exactly the answer to the why question we are looking for here. Seeing the policies as the means for guiding systems' actions towards behaviors to achieve optimal outcomes, the *Condition(s)* policy block refers to the actions level of the policy. Our focus here is on the higher level, that of the system's behavior. Focusing on a higher level, gives us the agility to find different policies as far as the actions are concerned, that provides the same functionality in terms of behavior; and the different policies we are looking for are those which serve the needs of the asset requestors as well. Achieving this goal we achieve a win-win negotiation outcome like the one described in Chefs-Orange scenario. The next four steps describe through a graphical representation the process to reach such an outcome.

Step 1: The simplistic domain model of Figure 5.1 presents only the concepts involved in the smartphone scenario of Section 5.3.2 and their relationships. It hides however their properties. In the smartphone example for instance, we assume that the concept Asset has a property named *Provided capability* and that the Asset instance named

[1]The age and experience is just two random conditions, that could be constraints in an authorization policy and are used as examples here; they have no significant meaning explaining the policy IBN process.

SMD has the provided capability property named *Tethering*. Thus, the policy rule R1 by denying access to SMD, it denies access to any of SMD's provided capability as well. The IBN process starts by dealing first with the restrictive rule's *Target* block $RestrictiveRule_T$ (i.e., restricted asset). Trying to broaden the negotiation space in order to find alternative policies that satisfy both negotiators, it breaks down the asset's properties and isolates the provided capabilities, (in smartphone example for instance it isolates Tethering capability from SMD) as shown in Figure 5.4 while adding them in a list named $PCL[n]$ as shown in Algorithm's 5.1 step 1.

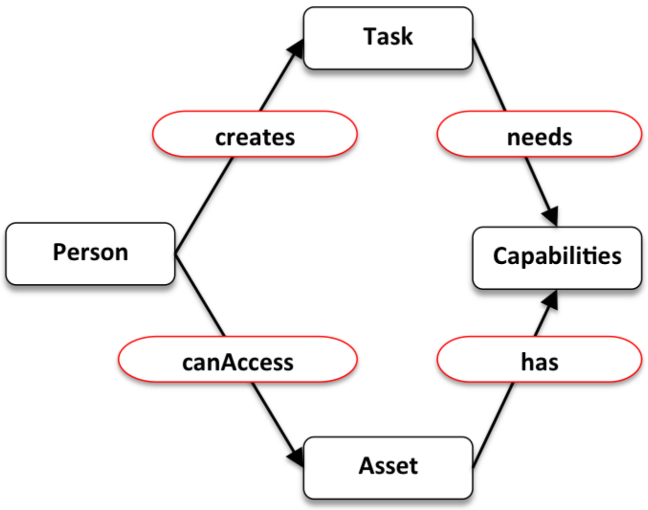


Figure 5.5: Ontology Modification: Step two

Step 2: Each Task of Figure 5.1 requires a set of capabilities in order to be served so it has a property, named *Required capability*. The Task instance of smart environment example (i.e., Email submission) has a number of required capabilities too, including *Tethering*. The second step of IBN algorithm deals with the Tasks issue; It breaks down their properties and isolates their required capabilities indicated as Task_{RC} in Algorithm in Algorithm's 5.1. In smartphone example for instance, it isolates the Tethering capability from task Email submission as shown in Figure 5.5. It adds those capabilities then in the required capabilities list $RCL[n]$ as shown in Algorithm's 5.1 step 2.

Step 3: Often the tasks' required capabilities might span outside the capabilities offered

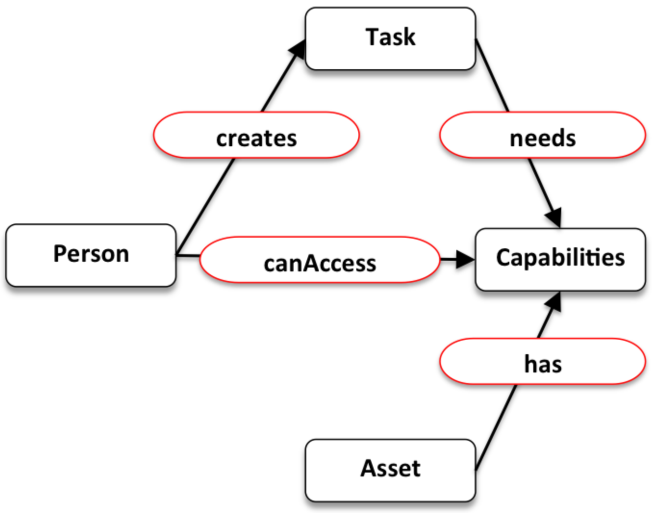


Figure 5.6: Ontology Modification: Step three

Algorithm 5.1: Policy IBN Algorithm

```
Input: restrictiveRule, Task, assetRequestor
1: add(restrictiveRule$_T$ProvidedCapability, PCL[n])
2: add(Task$_{RC}$, RCL[n])
3: CCL[n] = PCL[n]
4: CCL[n].retainAll(RCL[n])
5: for all CCL[n] do
6:     ruleRefined = createRefinedRule(restrictiveRule$_{ID}$+n, "Authorization", "", "assetRequestor", "canAccess", "RCL[n]")
7:     push(ruleRefined)
```

by one particular asset (e.g., a task might need to utilize capabilities provided by a set of assets). The third step of IBN algorithm, makes the matching between Asset's provided capabilities and Task's required capabilities. It matches this way the subset of the prohibited asset's properties that are needed for the implementation of the desired Task as shown in Figure 5.6. The asset requestor now can access a subset/subsystem of the asset that provides required capabilities for their task implementation. A third list, called common capabilities list $CCL[n]$ as shown in Algorithm's 5.1 steps 3 - 4 is created, which contains those matched capabilities.

Step 4: This step performs the *policy refinement*[2]. For all the common capabilities of $CCL[n]$ a new refined policy rule is created as shown in Algorithm's 5.1 steps 5 - 6. Step 6 describes the attributes of the refined policy rules. The refined policies' ID is the integer $ID = restrictiveRule_{\mathrm{ID}} + n$ while all the policies have *Authorization* as policy type. The asset requestor (in smartphone example that is P2) is the *Subject* block of the refined policy rules, which has as *Action* block a positive authorization (i.e., canAccess) while its *Target* block contains, the provided by the prohibited Asset and required by the desired Task capabilities of list $CCL[n]$ (in smartphone example that is Tethering). The condition policy block is empty. The CE refined policies are passed then to the asset owner for approval as shown in Algorithm's 5.1 step 7. Below we provide the refined policy *R1-Refined* of smartphone example.

```
Policy R1-Refined
if
   ( there is an asset A named SMD ) and
   ( the asset A has the capability C named Tethering ) and
   ( there is a person P named P2 )
then
   ( the person P canAccess the capability C ).
```

The asset owner P1 is in charge of confirming or not the replacement of policy R1 from the proposed policy R1-Refined. In the case of confirmation the refined policy is either stored permanently in Policy Repository replacing its predecessor or can only be enforced once and then be discarded as explained before. The successful completion of IBN leads the negotiating parties to a win-win negotiation, with the asset requestor get-

[2]Note that the term policy refinement herein refers to a different process than the policy refinement in [11], which describes the process of interpreting more general, business layer policies to more specific, system layer ones.

ting the task of *Email submission* served and the asset owner prohibiting any physical access to SMD.

5.6 Policy IBN Evaluation

The ideal evaluation of the behavior and effectiveness of IBN in policy refinement would be if applying it to human users operating for instance in an opportunistic network scenario sharing their assets, subject to authorization policies. Considering their own sharing constraints they would be responsible for the acceptance or rejection of the refined policies proposed by IBN.

5.6.1 Simulation Setup

Given that we lack the resources for such an experimentation we simulate this environment in Java. The simulation describes an asset sharing scenario in a small and short lived opportunistic network. In the scenario there are three basic concepts. The human users, their assets and the tasks they create similarly to the experiments of Chapter 2. The users are the asset owners and responsible for their sharing with the others through a policy-based system. Being eager for information consumption users create tasks which in order to be served, they need particular resources provided by the deployed assets. Thus, tasks are the entities that require capability while assets are the ones that provide capability in order for a task to be served. Often the task creators are not able to serve their tasks utilizing their asset resources and ask for support by users in the opportunistic network.

The opportunistic network scenario assumes 8 users. Each one of them owns one asset. There are three types of assets as many as the type of the tasks the users can create. Each asset type has the capability to serve a particular task type meeting its information requirements. As far as the asset sharing is concerned, it is managed through policies written by asset owners. Each user is responsible for whether to exclusively use their assets (following concerns regarding security, privacy and other issues) or sharing them with the others. We do not assume any spatial constraints in the simulated opportunistic network which means that all the users operate in a field in a distance where their devices have enough transmission/reception capability to cover communication range and share information with each other.

Moreover, one out of three asset types has a monolithic architectural design making it capable of serving only one particular task unlike the other two asset types, which are more capable devices and can operate as platforms of provided capabilities able to serve more than one task types.

To visualize better the simulated scenario think of the following vignette. Eight users are a group of people (i.e., network's users) hiking a mountain. The three types of assets might be a smartphone device such as the SMD described in previous sections, a music player equipped with transmitter/receiver and communication protocol capabilities able to communicate with other assets of the same type and a monolithic wearable pedometer device. The three possible tasks created by users are these of emailing, which requires internet connection provided by a smartphone device, music sharing which is served by portable music players capable of exchanging songs and playlists with other devices of the same type and that of steps counting served by the monolithic pedometer.

The IBN following the algorithm that was explained in the previous section is only capable to be applied on polylithic assets when strict policies are applied on (i.e., SMD and music player device). For the implementation of the step counting task the user needs to physically access a pedometer device. If the user that creates a steps counting task either does not own a pedometer device or any of the pedometer devices of the network are not sharable due to strict sharing policies, the policy IBN mechanism is unable to provide any refinement in this case.

The total number of created tasks is 100. They are created randomly by the eight users, which implies uneven number of tasks for each user[3]. Task types are also randomly created as do the types of the deployed/user owned assets. As it was mentioned before, the main objective of the IBN mechanism is the maximization of social welfare. To measure the effect of IBN on social welfare in our scenario we use as metrics the proportion of served and dropped tasks (the definition of served and dropped tasks is as described in detail in Chapter 2).

To have a complete picture of IBN effect on social welfare we experiment with three asset sharing models. The first is the strictest one and deals with very conservative, in terms of sharing policies, users where they do not share any of their devices with others in the network. In this case the asset sharing is set to 0% and the tasks created in the simulation can only be served if their creators' devices are capable to do so. In the second experiment we set the asset sharing to 25% namely 25% of the total devices are sharable and finally the last and most liberal case deals with 50% asset sharing. In all three experiments we measure the proportion of served and dropped tasks when: a) the IBN mechanism is deactivated **IBN OFF** and b) the IBN mechanism is activated **IBN ON**. For all six experimental cases

- *Asset sharing 0%:* 1) IBN OFF, 2) IBN ON Figure 5.8
- *Asset sharing 25%:* 1) IBN OFF, 2) IBN ON Figure 5.9
- *Asset sharing 50%:* 1) IBN OFF, 2) IBN ON Figure 5.10

we repeat each simulation instance 100 times averaging the measurements.

[3]Randomness in this simulation is achieved utilizing the generation of pseudorandom numbers of Java Random class. The class uses a 48-bit seed, which is modified using a linear congruential formula as described in [48]

5.6.2 Simulation RedLines

For simulating the acceptance or rejection of the refined policies (i.e., the relaxed policies provided by IBN) we use a mechanism called *RedLine*. We borrowed the term from RedLine from the worldwide used phrase "Red line", or "cross the red line", which means a figurative point of no return or a limit past which safety can no longer be guaranteed. Each user at the beginning of the simulation, randomly gets their RedLines settled, and that way they define their willingness to accept or reject the refined policies given their personal reasons. Given the complexity, in terms of assets capabilities and tasks requirements, we assume, it is difficult for opportunistic network's users to write fine-grained policies to define access control on every possible combination of them. In SMD case described in previous section for instance, the asset owner P1, being unable to cope with the complexity of matching SMD's capabilities and Email submission task's requirements he opts not to share any of SMD's capabilities with P2 following personal concerns. Hence, he simply express his constraints at higher level, setting access control policies at the assets level only. The RedLine mechanism, defines the distance between the asset owners' high-level authorization policies and their ”real" willingness to share their assets or subsets of them with their peers.

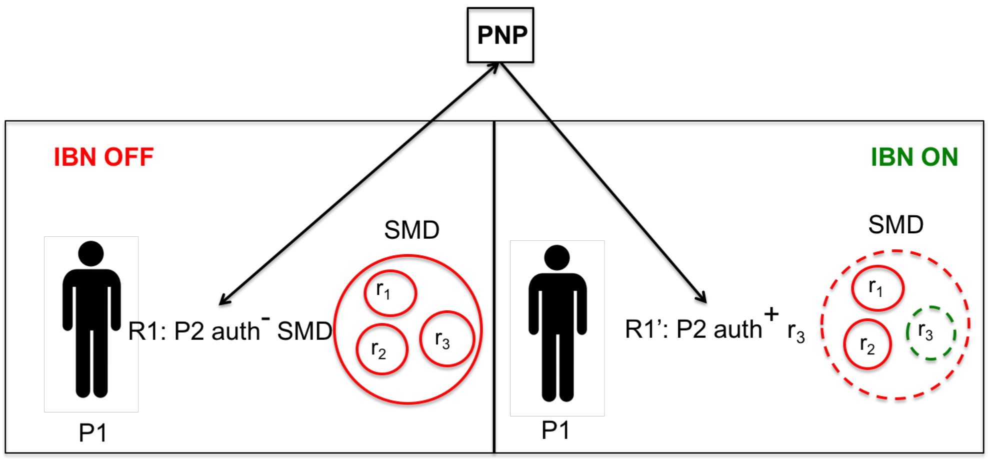


Figure 5.7: PNP: IBN OFF vs IBN ON

This assumption is depicted in Figure 5.7. On the left hand side where IBN is inactive, user P1 express his strict constraints, in terms of SMD sharing with P2 through high-level policy rule R1, according to which the requestor user P2 is not allowed to access it and as a consequence he forbids access to any of SMD's subsystems (namely resources r_1, r_2 & r_3 and their provided capabilities). The PNP, activating IBN mechanism as shown in the right hand side of Figure 5.7 proposes a finer-grained rule R1' according to which the requestor user R2 can access (as the dotted line circles indicate) the necessary SMD subsystem (namely resource r_3 and its provided capability) in order to get his task served. The intention of P1 in terms of approving or rejecting the refined R1' is simulated by users' RedLines mechanism.

IBN mechanism, as mentioned before, attempts to lower barriers, through policy refinement, in order to establish better collaboration through asset sharing (i.e., increase the overall number of served tasks and thus increase the social welfare) while maintaining the compromise from the asset owners point of view at the same levels. Those users that their RedLines are more relaxed compared to their policies represent those who believe that they do not compromise any of their concerns expressed through their strict sharing policies and proceed with accepting the refined ones.

5.6.3 Simulation Results

The simulation results are presented through Figures 5.8, 5.9 & 5.10, and the error bars on clustered column charts represent (+ / -) 1 Standard Deviation. In all three experiments, when the IBN mechanism is activated the social welfare in terms of task implementation is higher as expected. As it is shown the effectiveness of IBN is higher in strict environments and decreases as moving to more liberal ones. The IBN OFF columns indicate that only 32% of the total tasks are served meaning that given the need for information of the users only the one third of it can be served by their own resources. For the same setup having IBN ON the proportion of served tasks increases to 50%.

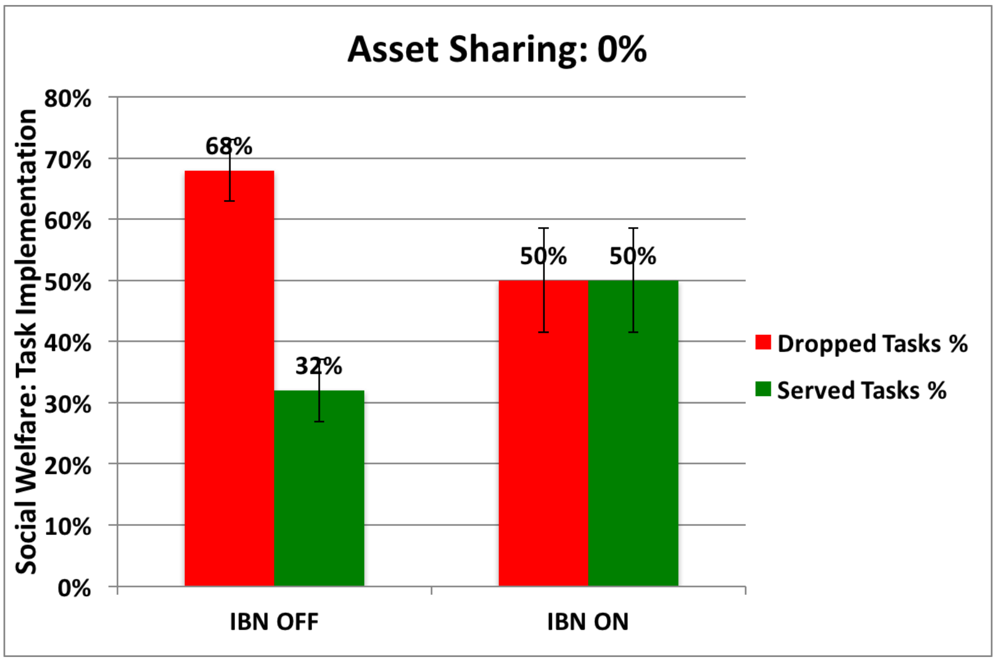


Figure 5.8: IBN effect on social welfare: Asset Sharing 0%

More in details, in first and most strict case Figure 5.8, the margin between dropped and served tasks when IBN is inactive is 36 percentage units. When IBN is active the proportion of dropped and served tasks is even. In the second experiment Figure 5.9 the served tasks proportion outperforms the dropped tasks' whether the IBN is active or not. With IBN ON however, the proportion of served tasks is 8 percentage units more compared to when IBN is OFF.

Same trend in the last and most liberal case where half of the assets in the opportunistic network are shared with the network's users. The margin here between IBN ON and IBN OFF cases almost disappears with IBN ON case performing slightly better with 2 percentage units. From the results in Figure 5.10 it is also inferred that for the simulated opportunistic network environment, if the asset sharing ratio is higher than 50% the IBN mechanism has not much to offer, while it is a useful and effective tool in promoting collaboration through asset sharing, for stricter (in terms of asset sharing) environments.

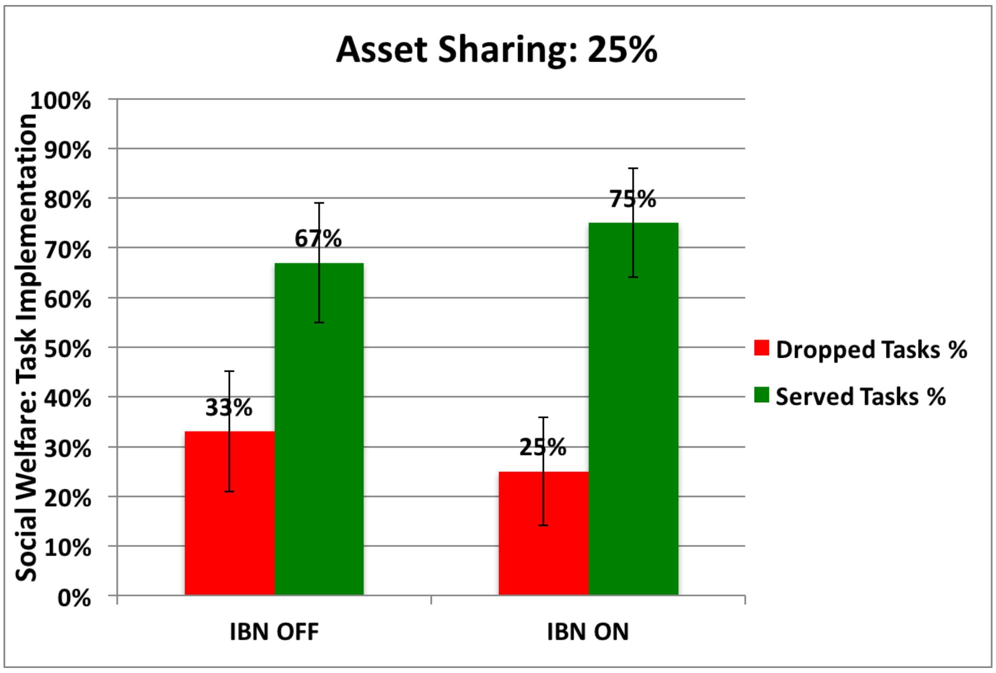


Figure 5.9: IBN effect on social welfare: Asset Sharing 25%

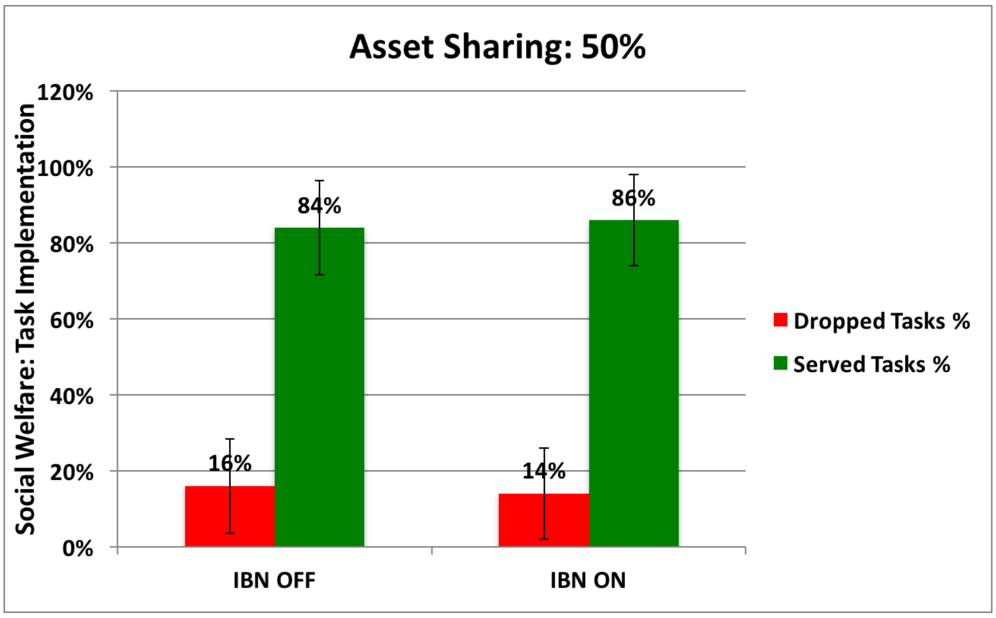


Figure 5.10: IBN effect on social welfare: Asset Sharing 50%

Chapter 6

Conclusion & Future Work

Resource sharing is an important but complex problem to be solved. The problem is exacerbated in a dynamic coalition context, due to multi-partner constraints (imposed by security, privacy and general operational issues) placed on the resources. Policy-Based Management Systems have been proposed as a suitable paradigm to reduce this complexity and provide a means for effective asset sharing. The overall complexity for developing an effective resource sharing model is increased when dealing with a decentralized management approach that enables non IT expert administrators to interfere with resource sharing configurations while operating near or at the organizational edge.

The overarching problem that this thesis deals with, is the development of PBMS techniques and technologies that will allow in a dynamic and transparent way, users that operate in collaborative environments to share their assets through high-level policies. To do so it focused on four sub-problems each one of which relates to different PBMS aspects and was researched separately and presented through chapters 2 to 5.

The first piece of research work dealt with effective ways of resources' sharing, through high-level policies in collaborating and dynamic environments. The novel resource sharing model, called team-centric sharing model was investigated under the light of smart environment resource sharing, utilized to support responders that operate in unexpected and turbulent emergency operation.

The second piece of research work proposed a policy conflict analysis mechanism that provides the means for the development of a transparent – in terms of user interaction – model for maintaining unconflicted PBMS. Focusing on multi-partner collaborating environments where the likelihood for conflicting policies is relatively high the proposed mechanism has been designed for efficient – in terms of time complexity – policy conflict analysis. Finally, the piece of research work that concludes this thesis dealt with an interest-based policy negotiation mechanism for enhancing asset sharing while promoting collaboration in coalition environments. We believe it would be useful to propose an IBN approach given that it promotes collaboration when compared with more traditional negotiation approaches such as position-based negotiations and measure its effectiveness, in strict (in terms of asset sharing policies), collaborating environments.

We do not consider the proposal of the user-friendly policy language, presented in Section 3 as one of our key contributions; it mostly provides the enabling technology to our three previous contributions. It looked into a policy language, suitable for distributed policy-based system administration by users operating near or at the organizational edge. We illustrated the language's expressiveness and user friendliness writing policies for sharing resources under the spectrum of an army related scenario.

The following sections provide an overview of this thesis contributions, point out potential future directions and conclude this thesis with some remarks.

6.1 Contribution Overview

Team-drive Asset Sharing: We presented the team-centric sharing model; a novel, policy-based asset sharing scheme inspired from military operations, based on the *edge C2* model. This asset sharing approach shifts decision making power regarding assets' access control to the edge of organizations [53], thus allowing for more dynamic formation of teams and assets sharing patterns to emerge. Although the existing asset sharing models cope with the problem of resources and services sharing among collaborative organizations in a secure and confidential manner, they do not address directly highly dynamic environments. Thus, they are likely to fail or encounter difficulties in being applied to those cases due to the extra overhead that is needed in order for all of them to comply with dynamic and frequent environmental changes. Differently, the team-centric sharing model due to its event-driven formation nature does not present any overhead regardless the frequency of the environmental changes.

We formalized, evaluated and compared the novel proposed asset sharing model, with the traditional approach investigating its impact on policy enabled Multi-Sensor Task Allocation protocol. In particular, with this piece of research work, we found that while the traditional ownership model allows slightly better performance, the difference is only marginal, so a team-sharing model offers a viable alternative sharing approach for dynamic and collaborative scenarios.

CE-based Policy Conflict Analysis: Responding to the characteristics of the scenarios this thesis looks into, we presented a policy conflict analysis based on a Controlled English approach. Although the problem of maintaining unconflicted PBMS has been thoroughly researched the last two decades, which as a result enriched the literature with many alternative approaches, none of them addresses directly environments that this thesis deals with. Coping with highly dynamic and heterogeneous – in terms of policy authoring authorities – environments, which as a consequence implies a potentially highly conflicting PBMS, we presented a time efficient policy conflict analysis mechanism. Moreover, the environments that this thesis focuses on are populated by non IT expert users so a more transparent approach compared to the state-of-the-art is required.

Exploiting the semantics of CE and combining it with its hybrid ways of reasoning, we were armed with a set of tools for developing a time efficient, discrete and sequentially executed policy conflict analysis algorithms. The superior to naive (in terms of policies' comparison) approaches, CE-based, discipline independent and able to be used in generic PBMS middleware, policy conflict analysis was evaluated and presented as well.

Interest-based Policy Negotiation: In summary the proposed IBN mechanism provides an effective policy negotiation tool for revising asset sharing policies in dynamic, multi-party environments. Although there is rich literature on negotiation protocols in autonomous MAS, there is very limited and no mature work done on policy negotiation. We strongly believe that policy negotiation is an important feature to have, interfacing seamlessly with PBMS and existing policies, because this is where the core component of the system's management logic resides. The proposed IBN negotiation unlike the traditional PBN approaches tackles the problem of negotiation by focusing on “why negotiate for" rather than on “what to negotiate for", aiming to lead negotiating parties to win-win solutions, promoting collaboration. We concluded that policy IBN is a useful and effective tool in terms of promoting collaboration through asset sharing, for strict in terms of sharing environments.

6.2 Future Research Directions

This section presents ways and ideas in which the work in this thesis can be extended and further investigated. Potential future directions are summarized below:

CE Policy Language: Although we examined the expressiveness of CE policy language and we claimed that it constitutes a user-friendlier approach compared to predecessors, we did not perform any formal evaluation to justify our claim. We are planning to conduct human-led experiments, with undergraduate university students, to compare CE policy language with other formal policy representations in terms of in terms of their user-friendliness. A measurable, will be the time needed for users to decently learn how to express policy rules with CE and other representations, while an additional, interest point to look into in such an experiment is to measure the readability of CE expressed policies and compare it with others when users have zero training.

PBMS Development: We proposed herein components that can be used in PBMS development. We did not develop a final prototype that can apply policy-based management. It is not very research oriented work but in the near future we plan to integrate our work with the WPML framework, developed by collaborators in IBM Watson Research Lab. By merging the our technologies with their framework, we intent to achieve a more intelligent, efficient and transparent PBMS that fits better in dynamic and distributed environments.

Interest-based Policy Negotiation: Moreover, there are plans for extending the IBN steps with regards to broadening the negotiation space considering heuristics related to users and their characteristics such as their team affiliation, that can improve IBN's effectiveness through sharing assets and their provided service horizontally (e.g. inner-team) unlike the current vertical (i.e., user-to-user) approach. The policy IBN work is going to be implemented and integrated the following months in a big "Capstone" demo, which will summarize technologies that emerged from ITA project the last decade.

Policy Refinement: We have recently started, looking into another issue related to policy-based management, this of Policy Refinement, which is still in progress. The policy refinement problem as addressed in [82, 23] describes the methodologies for building policy hierarchies so that to refine abstract, high-level policies (like the ones this thesis mostly deals with) into more technical, fine-grained, executable policies. We believe that, by utilizing CE as the means for defining both, the managed system and the policies that govern it and its semantics and the ability to build hierarchies of concepts will be helpful in developing a transparent and automatic mechanism for concepts decomposition and thus policy refinement.

Bibliography

[1] CIM policy model white paper. `http://www.dmtf.org/sites/default/files/standards/documents/DSP0108.pdf`, 2003. Accessed: 29/09/2015.

[2] ABCA: Coalition operations handbook. *ABCA Publication 332*, 2008.

[3] CIM Simplified policy language (CIM-SPL). *DMTF*, 2009.

[4] D. Agrawal, S. Calo, J. Giles, Kang-Won Lee, and D. Verma. Policy management for networked systems and applications. In *Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International Symposium on*, pages 455–468, May 2005.

[5] D. Agrawal, J. Giles, K. Lee, and J. Lobo. Policy ratification. In *In POLICY 2005: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY2005*, pages 223–232. IEEE Computer Society, 2005.

[6] D. Agrawal, J. Giles, K. W. Lee, and J. Lobo. Policy ratification. volume 2005 of *6th IEEE International Workshop on Policies for Distributed Systems and Networks*.

[7] D. Agrawal, J. Giles, Kang-Won Lee, K. Voruganti, and K. Filali-Adib. Policy-based validation of SAN configuration. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, pages 77–86, June 2004.

[8] D.S. Alberts, R.K. Huber, and J. Moffat. NATO NEC C2 Maturity Model. *CCRP*, 2010.

[9] F. Bai and A. Helmy. A survey of mobility models in wireless adhoc networks. In *(Chapter 1, Wireless Ad Hoc Networks. Kluwer Academic*, 2006.

[10] J. Bailey, G. Papamarkos, A. Poulovassilis, and P. T. Wood. An event-condition-action language for XML. In *Web Dynamics*, pages 223–248. Springer Berlin Heidelberg, 2004.

[11] A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo. A goal-based approach to policy refinement. *Proceedings - Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY 2004*, pages 229–239, 2004.

[12] A.K. Bandara, E.C. Lupu, and A. Russo. Using event calculus to formalise policy specification and analysis. In *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, pages 26–39, June 2003.

[13] J. Barron, S. Davy, and B. Jennings. Conflict analysis during authoring of management policies for federations. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pages 1180–1187, May 2011.

[14] A. Bernstein and E. Kaufmann. GINO - a guided input natural language ontology editor. 2006.

[15] R. Bhatti, A. Samuel, M.Y. Eltabakh, H. Amjad, and A. Ghafoor. Engineering a policy-based system for federated healthcare databases. *IEEE Transactions on Knowledge and Data Engineering*, 19(9):1288–1304, 2007.

[16] C. Bisdikian, M. Sensoy, T.J. Norman, and M.B. Srivastava. Trust and obfuscation principles for quality of information in emerging pervasive environments. pages 44–49, 2012.

[17] A. Brandenburger and A. Nalebuff. Co-opetition. *New York: Doubleday*, 1997.

[18] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A simple network management protocol (SNMP). `http://www.ietf.org/rfc/rfc1157`, 1990. Accessed: 29/09/2015.

[19] M. Charalambides, P. Flegkas, G. Pavlou, Javier Rubio-Loyola, A.K. Bandara, E.C. Lupu, A. Russo, N. Dulay, and M. Sloman. Policy conflict analysis for diffserv quality of service management. *Network and Service Management, IEEE Transactions on*, 6(1):15–30, March 2009.

[20] M. Charalambides, P. Flegkas, G. Pavlou, Javier Rubio-Loyola, A.K. Bandara, E.C. Lupu, A. Russo, M. Sloman, and N. Dulay. Dynamic policy analysis and conflict resolution for diffserv quality of service management. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 294–304, April 2006.

[21] V. Cheng, P. Hung, and D. Chiu. Enabling web services policy negotiation with privacy preserved using XACML. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2007.

[22] M. Comuzzi and C. Francalanci. Agent-based negotiation in cooperative processes: Automatic support to underwriting insurance policies. *ACM International Conference Proceeding Series*, 60:121–129, 2004.

[23] R. Craven, J. Lobo, E. Lupu, A. Russo, and M. Sloman. Decomposition techniques for policy refinement. In *Network and Service Management (CNSM), 2010 International Conference on*, pages 72–79, Oct 2010.

[24] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder policy specification language. In *Policies for Distributed Systems and Networks*, volume 1995 of *Lecture Notes in Computer Science*, pages 18–38. Springer Berlin Heidelberg, 2001.

[25] N. Damianou, N. Dulay, E. Lupu, M. Sloman, and T. Tonouchi. Tools for domain-based policy management of distributed systems. In *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pages 203–217, 2002.

[26] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. In *Selected Papers of the Sixth International Workshop on Software Specification and Design*, 6IWSSD, pages 3–50. Elsevier Science Publishers B. V., 1993.

[27] Mott. Dave and al. et. Hybrid rationale and controlled natural language for shared understanding. *The 6th Annual Conference of Knowledge Systems for Coalition Operations.*, 2010.

[28] S. Davy, B. Jennings, and J. Strassner. Application domain independent policy conflict analysis using information models. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 17–24, April 2008.

[29] S. Davy, B. Jennings, and J. Strassner. Efficient policy conflict analysis for autonomic network management. In *Engineering of Autonomic and Autonomous Systems, 2008. EASE 2008. Fifth IEEE Workshop on*, pages 16–24, March 2008.

[30] S. Davy, B. Jennings, and J. Strassner. Using an information model and associated ontology for selection of policies for conflict analysis. *Policies for Distributed Systems and Networks, IEEE International Workshop*, pages 82–85, 2008.

[31] M.M. Delaney, A. Foroughi, and W.C. Perkins. An empirical study of the efficacy of a computerized negotiation support system (NSS). *Decision Support Systems*, 20(3):185–197, 1997.

[32] N. Dulay, E. Lupu, M. Sloman, and N. Damianou. A policy deployment model for the ponder language. In *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, pages 529–543, 2001.

[33] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (common open policy service) protocol. `http://www.ietf.org/rfc/rfc2748`, 2000. Accessed: 29/09/2015.

[34] S. Ellison, C. Dohrmann. Public-key support for group collaboration. *ACM Transactions on Information and System Security*, 6(4):547–565, 2003.

[35] Peyman Faratin, Carles Sierra, and Nick R. Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3 - 4):159 – 182, 1998. Multi-Agent Rationality.

[36] B.S. Firozabadi, M. Sergot, A. Squicciarin, and E. Bertino. A framework for contractual resource sharing in coalitions. In *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop*, pages 117–126, June 2004.

[37] R. Fisher et al. *Getting to yes: Negotiating an agreement without giving in; 2nd repr. ed.* Random House Business Books, London, 1999.

[38] T. Frisanco, P. Tafertshofer, P. Lurin, and R. Ang. Infrastructure sharing and shared operations for mobile network operators from a deployment and operations view. In *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pages 129–136, 2008.

[39] M. Gomez, A. Preece, M.P. Johnson, G. De Mel, W. Vasconcelos, C. Gibson, A. Bar-Noy, K. Borowiecki, T. La Porta, D. Pizzocaro, H. Rowaihy, G. Pearson, and T. Pham. An ontology-centric approach to sensor-mission assignment. 2008.

[40] R. Govindan, T. Faber, J. Heidemann, D. Estrin, et al. Ad hoc smart environments. In *Proceedings of the DARPA/NIST Workshop on Smart Environments*, 1999.

[41] H. Hamed and E. Al-Shaer. Taxonomy of conflicts in network security policies. *Communications Magazine, IEEE*, 44(3):134–141, March 2006.

[42] G. Hart, M. Johnson, and C. Dolbear. Rabbit: Developing a control natural language for authoring ontologies. 2008.

[43] R. Hayes and D. Alberts. Power to the edge: Command and control in the information age. *CCRP*, 2003.

[44] R. Hayes and D. Alberts. Power to the edge: Command and control in the information age. *CCRP*, 2003.

[45] I.J. Jang, W. Shi, and H.S. Yoo. Policy negotiation system architecture for privacy protection. *Proceedings - 4th International Conference on Networked Computing and Advanced Information Management, NCM 2008*, 2:592–597, 2008.

[46] N.R. Jennings et al. Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.

[47] L. Kagal, T. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, POLICY ’03, pages 63–, Washington, DC, USA, 2003. IEEE Computer Society.

[48] D. Knuth. The art of computer programming. volume 2, page 672. Addison-Wesley Professiona.

[49] S. Kraus, O. Shehory, and G. Taase. Coalition formation with uncertain heterogeneous information. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’03, pages 1–8, New York, NY, USA, 2003. ACM.

[50] Sarit Kraus. *Strategic Negotiation in Multiagent Environments*. MIT Press, Cambridge, MA, USA, 2001.

[51] T. Kuhn. An evaluation framework for controlled natural languages. In *Proceedings of the 2009 Conference on Controlled Natural Language*, CNL'09, pages 1–20, Berlin, Heidelberg, 2010. Springer-Verlag.

[52] T. Kuhn. A survey and classification of controlled natural languages. *Comput. Linguist.*, 40(1):121–170, March 2014.

[53] T.A. Leweling and M. E. Nissen. Hypothesis testing of edge organizations: Laboratory experimentation using the ELICIT multiplayer intelligence game. In *In Proc. of ICCRTS*, June 2007.

[54] Q. Li and A. Vijayalakshmi. Concept-level access control for the semantic web. In *Proceedings of the 2003 ACM Workshop on XML Security*, XMLSEC 2003, pages 94–103, New York, USA, 2003. ACM.

[55] J. Lobo, R. Bhatia, and S. Naqvi. A policy description language. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 291–298, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[56] C. Loebecke, P. C. Van Fenema, and P. Powell. Co-opetition and knowledge transfer. *SIGMIS Database*, 30(2):14–25, 1999.

[57] X. Lu, Y. C. Chen, I. Leung, Z. Xiong, and P. Li. A novel mobility model from a heterogeneous military MANET trace. 7th International Conference on Ad-hoc, Mobile and Wireless Networks. 2008.

[58] E. C. Lupu and M. Sloman. Conflicts in policy-based distributed systems management. *IEEE Trans. Softw. Eng.*, 25(6):852–869, November 1999.

[59] L. Lymberopoulos, E. Lupu, and M. Sloman. An adaptive policy-based framework for network services management. *Journal of Network and Systems Management*, 11(3):277–303, 2003.

[60] J. Mahaffey and T. Skaar. Observations on the dissemination of isr data employing network-enabled capabilities in the coalition environment, 2005.

[61] Osborne Martin and Rubinstein Ariel. A course in game theory. *The MIT Press*, 1994.

[62] A. Medina, G. Gursun, P. Basu, and I. Matta. On the universal generation of mobility models. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*, pages 444–446, Aug 2010.

[63] J.D. Moffett and M.S. Sloman. Policy hierarchies for distributed systems management. *Selected Areas in Communications, IEEE Journal on*, 11(9):1404–1414, Dec 1993.

[64] B. Moore. Policy core information model (PCIM) extensions. United States, 2003. RFC Editor.

[65] Thomas P. Moran. *Design Rationale: Concepts, Techniques, and Use*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1996.

[66] D. Mott. Summary of controlled english. *ITACS*, 2010.

[67] E. Mykoniati, C. Charalampous, P. Georgatsos, T. Damilatis, D. Goderis, P. Trimintzios, G. Pavlou, and D. Griffin. Admission control for providing QoS in diffserv IP networks: The TEQUILA approach. *IEEE Communications Magazine*, 41(1):38–44, 2003.

[68] C. Nita-Rotaru and N. A. Li. Framework for role-based access control in group communication systems. In *In Proc. of International Workshop on Security in Parallel and Distributed Systems*, 2004.

[69] OASIS. eXtensible Access control markup language (XACML) version 3.0, 2013.

[70] C. Parizas, D. Pizzocaro, A. Preece, and P. Zerfos. Managing ISR sharing policies at the network edge using controlled english. *In proceedings of Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IV*, 2013.

[71] C. Parizas and P. Zerfos. CE-based policies conflict analysis for sensor services management in information fabric. Palisades, NY USA, 2013. Proceedings of the 1st Annual Fall Meeting of ITA.

[72] P. Pasquier et al. An empirical study of interest-based negotiation. *Autonomous Agents and Multi-Agent Systems*, 22(2):249–288, 2011.

[73] G. Pearson and T. Pham. The challenge of sensor information processing and delivery within network and information science research. volume 6981, pages 698105–698105–9. SPIE.

[74] T. Pham and G. Cirincione. Sensor data and information sharing for coalition operations, 2012.

[75] T. Pham, G. Cirincione, D. Verma, and G. Pearson. Intelligence, surveillance, and reconnaisance fusion for coalition operations. In *Proc 11th International Conference on Information Fusion*, 2008.

[76] T. Pham, G. Pearson, F. Bergamaschi, and S. Calo. The ITA sensor fabric and policy management toolkit, 2011.

[77] D. Pizzocaro, A. Preece, F. Chen, T. L. Porta, and A. Bar-Noy. A distributed architecture for heterogeneous multi sensor-task allocation. 7th IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS'11.

[78] A. Preece, D. Braines, D. Pizzocaro, and C. Parizas. Human-machine conversations to support mission-oriented information provision. *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, pages 43–49, 2013.

[79] A. Preece, C. Gwilliams, C. Parizas, C. Pizzocaro, J. Bakdash, and D. Braines. Conversational sensing. In *SPIE 9122, Next-Generation Analyst II*, Baltimore, MD, USA, 2014.

[80] A. Preece, D. Pizzocaro, D. Braines, and D. Mott. Tasking and sharing sensing assets using controlled natural language. In *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR III*, volume SPIE Vol 8389, 2012.

[81] I. Rahwan et al. Argumentation-based negotiation. *Knowledge Engineering Review*, 18(4):343–375, 2003.

[82] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, and G. Pavlou. A methodological approach toward the refinement problem in policy-based management systems. *Comm. Mag.*, 44(10):60–68, October 2006.

[83] T. Samak and E. Al-Shaer. Fuzzy conflict analysis for qos policy parameters in diffserv networks. *Network and Service Management, IEEE Transactions on*, 9(4):459–472, December 2012.

[84] T.W. Sandholm. Distributed rational decision making. *Multiagent Systems (ed. G. Weiss) MIT Press*, pages 201–258, 1999.

[85] R. Schwitter. Controlled natural language as interface language to the semantic web. pages 1699–1718, 2005.

[86] K.E. Seamons et al. Requirements for policy languages for trust negotiation. *3rd International Workshop on Policies for Distributed Systems and Networks, POLICY 2002*, pages 68–79, 2002.

[87] M. Sensoy, T. J. Norman, W. W. Vasconcelos, and K. Sycara. OWL-POLAR: Semantic policies for agent reasoning. In *The Semantic Web - ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*, pages 679–695. Springer Berlin Heidelberg, 2010.

[88] A. Simon and M. James. The agile organization. *CCRP*, 2005.

[89] M. Sloman and E. Lupu. Policy specification for programmable networks. In *Active Networks*, volume 1653 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin Heidelberg, 1999.

[90] J. Strassner. DEN-ng: achieving business-driven network management. In *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, pages 753–766, 2002.

[91] K. Su, J. Li, and H. Fu. Smart city and the applications. pages 1028–1031, 2011.

[92] J.-F. Tang and X.-L. Xu. An adaptive model of service composition based on policy driven and multi-agent negotiation. *Proceedings of the 2006 International Conference on Machine Learning and Cybernetics*, 2006:113–118, 2006.

[93] M. Tilbrook and R. Schwitter. Controlled natural language meets the semantic web. In *Australasian Language Technology Workshop 2004*, volume 2, 2004.

[94] G. Tonti, J. Bradshaw, R. Jeffers, R. Montanari, N. Suri, and A. Uszok. Semantic web languages for policy representation and reasoning: A comparison of kaos,

rei, and ponder. In *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 419–437. Springer Berlin Heidelberg, 2003.

[95] William Umiker. Collaborative conflict resolution. *The Health Care Manager*, 15(3), 1997.

[96] D. C. Verma, T. Brown, and C. Ortega. Management of coalition sensor networks. volume 7694 of *Ground/Air Multi-Sensor Interoperability, Integration, and Networking for Persistent ISR*.

[97] D.C. Verma. Simplifying network administration using policy-based management. *IEEE Network*, 16(2):20–26, 2002.

[98] E Vranes. The definition of norm conflict in international law and legal theory. In *European Journal of International Law*, pages 395–418, 2006.

[99] D.N. Walton and E.C. Krabbe. Commitment in dialogue: Basic concepts of interpersonal reasoning. *Albany, New York: SUNY Press*, 1995.

[100] M. Weiser, R. Gold, and J.S. Brown. Origins of ubiquitous computing research at PARC in the late 1980s. *IBM Systems Journal*, 38(4):693–695, 1999.

[101] L. WonYoung, S. HeeSuk, and C. TaeHo. Modeling of policy-based network with SVDB. 3397:323–332, 2005.

[102] D. Wood. Developer's guide to the watson policy management library (WPML). 2009. Accessed: 29/09/2015.

[103] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10:115–152, 6 1995.

[104] J. Wright, C. Gibson, F. Bergamaschi, K. Marcus, R. Pressley, G. Verma, and G. Whipps. A dynamic infrastructure for interconnecting disparate ISR/ISTAR assets (the ITA sensor fabric). In *IEEE/ISIF Fusion 2009 Conference*, 2009.

[105] P. Xue, D. Mott, D. Braines, S. Poteet, A. Kao, and C. Giammanco. Information extraction using controlled english to support knowledge-sharing and decision-making, 2012.

[106] R. Yavatkar, K. Pendarakis, and R. Guerin. A framework for policy-based admission control. `http://www.ietf.org/rfc/rfc2753.txt`, 2000. Accessed: 29/09/2015.

[107] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. 2010.

[108] K. Zeilenga. Lightweight directory access protocol (LDAP): Technical specification road map. `http://www.ietf.org/rfc/rfc4510`, 2006. Accessed: 29/09/2015.