Applying Digital Rights Management to Corporate Information Systems

Ziyi Su

N° d'ordre 2012ISAL0027 Année 2012

Thèse

Application de gestion des droits numériques au système d'information d'entreprise

Présentée devant
L'institut national des sciences appliquées de Lyon

Pour obtenir
Le grade de docteur

Formation doctorale : Informatique
École doctorale : Informatique et Mathématique de Lyon

Par
Ziyi SU

Soutenue le 22 mars 2012 devant la Commission d'examen

Jury MM.

Rapporteur	Prof. Ernesto DAMIANI
Rapporteur	Dr. Khalid BENALI
	Dr. Agnès FRONT
	Prof. Chirine GHEDIRA-GUEGAN
	Prof. Bruno VALLESPIR
	Prof. Frédérique BIENNIER

Laboratoire de recherche : Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)

Application de gestion des droits numériques au système d'information d'entreprise

Résumé

La sécurité bout-en-bout vise à protéger un actif (l'information et le processus, e.g. Services Web) de l'étape de la création à la démolition. Basé sur l'analyse des caractéristiques du system DRM, des systems contrôle d'accès et du system collaboratif , nous avons proposé un «système de contrôler l'utilisation collaboratif» pour cette tâche.

Le primier parti de notre travaille concerne une analyse des fonctionnalités et des composants dans un système de contrôler l'utilisation collaboratif. Un modèle de politique de contrôle d'utilisation qui intégrè les éléments de sécurité traditionnel et la fondation.

Nous avons défini une syntaxe concise et une sémantique formelle pour ce modèle politique et nous proposons une base de vocabulaire qui recueille commun est des facteurs de sécurité. Avec cette base de vocabulaire, notre modèle de collaboration contrôler l'utilisation est conscient des questions de sécurité conventionnelle.

Pour applicer dans les contextes collaboratif, nous proposons une méthode d'agrégation basé sur une «algèbre intégration», qui presente le raisonnement pour la co-autoriser un droit. Nous proposons un algorithmes pour analyser des processus d'affair (par exemple processus en WS-BPEL), pour decider lesquels politique doivent être intégré.

En ce qui concerne l'application, nous proposons une architecture de mise en œuvre. Nous construisons un moteur de négociation avec le outil «SUN XACML implementation», un moteur d'agrégation s'appuyant sur JAVA DOM et JDOM et et une composante d'analyse de contexte. Nous avons testé les performances de ces composants.

Mots-Clés: sécurité – droits numériques – politique de sécurité – architecture - algorithme - collaborative – sémantique – combinator de rules – gestion de context – implementation – performance

Applying Digital Rights Management to Corporate Information Systems

Abstract

To fit the globalised economical environment, enterprises, and mostly SMEs, have to develop new networked and collaborative strategies, focusing on networked value creation (instead of the classical value chain vision), fitting the blue ocean context for innovative products and service development. Such collaborative networks are by now often based on trusted and well known communities. Developing large scale networked and collaborative strategies involve increasing both enterprise and information system agility and interoperability in order to allow their interconnection. This requires paying attention on an end-to end security and on the way information and process are used during their full life-cycle. As traditional security approaches and methodologies provide only an “instant” and rather static protection, they do not fit the dynamicity nor the life-cycle long protection constraints involved by such collaborative organisation. To overcome this limit, we propose to adapt the Digital Right Management approach (first defined for multimedia contents) to collaborative information systems. After proposing a semi-distributed architecture used to manage usage rights, we propose a security policy model including both usage rights and related obligations. This leads us to extend the security policy descriptions, including a dedicated syntax and semantics to model both policy organisation, usage and obligations before paying attention on the “collaborative environment constraints”. Paying attention on the way collaborative organisations are set and evolve, we have proposed an integration algebra to manage the way security and usage policies are composed depending on the way partners join and quit the collaborative context. This composition process and integration algebra analyse the collaborative business processes to identify the way policies are composed and negotiated. Lastly, we implement parts of our architecture to validate our proposals, mostly regarding the negotiation engine (using «SUN XACML implementation»), the aggregation engine (built upon JAVA DOM et JDOM) and a context analysis component. As to implementation, we have proposed an arthitecture for the end-to-end security management, developed the ‘context management’, ‘Policy Decision Point’, ‘Policy Gathering Point’ components and presented the performance testing results.

Key words: security – ditital right management – security policy – architecture – algorithm – collaborative information system – semantic – rule combinator – context management – implementation – performance

Acknowledgments

I would like first of all to thank my supervisor Professor frédérique BIENNIER for having led me to the road of pursuit for knowledge in the frontier of computer science and information technology. She has passed down hand-to-hand experiences to me in the way of scientic thinking, research work origanizing and opinion presenting, with so much efforts and so much patient. Her enlightenment is like a beacon for young researchers like me, in our life-long quest for the holy grail of information science.

Supports from my family also give me much strength when I feel frustrated or bewildered. It soothe the pressure that I should face as a scientific researcher. Understanding and encouragement from my parents, my wife and my relatives and friends are I treasure.

My gratitude is heartful and enormous for the scholars who have put time and energy in this thesis. Comments from professor Ernesto DAMIANI and Dr. Khalid BENALI are so important that they spot the exact weaknesses scattered in the script and give so partinent advices for the improvement. The time, efforts, and opinions given by Dr. Agnès FRONT, professor Chirine GHEDIRA-GUEGAN and professor Bruno VALLESPIR are also key elements for the fulfillment of this thesis.

Applying Digital Rights Management to Corporate Information Systems

January 20, 2012

Contents

Chapter 1

Introduction

Recent changes in economy imposes new agility requirements to enterprises, e.g. the capability to respond to changes (client request, technology or methods evolvement, supplier management) [115], to adapt to structural changes [177] and to lean manufacturing strategy [309], as well as the new emerging service economic [295], etc. To face these challenges, new enterprise strategies rely more and more on inter-organizational collaborations. This can be achieved by connecting and integrating services among participants to offer final services or products for client is of great value (coalitions, supply/service chain, virtual enterprise, Cloud application, etc.).

As such, an exponential growth of innovative, pervasive services ecosystem is expected over the next few years. These ecosystems will rely on software services, which span multiple organizations and providers. Such dynamic service chain organization will provide agile support for business applications, government, or simply end user. In the services ecosystem, participants exchange value by (co-)providing and consuming information and processes that we'll later call "assets" [36]. Information, in its most restricted technical sense, is an ordered sequence of symbols' [193], which, in our perspective, provides the description of a fraction of the existence by itself (without inputs or outputs). We'll later consider information as "data" used by processes. 'Process' describes an act or a series of acts that take something (e.g. information) as input and produce outputs or leads to change the system state. We'll later consider services as a particular process implementation. As we focus on collaborative and distributed systems, used to set Collaborative Virtual Organisation, the two major technical conundrums are 'interoperability' and 'security'.

Interoperability refers to the capability allowing different systems to work together, communicating and exchanging information. This requires being able to process and understand information (both syntactically and semantically) to achieve a common business goal. This means that interoperability requirements must be taken into account at both technical, conceptual and business levels. The Service Oriented Architecture (SOA) fits the "technical interoperability" as it provides standardized interfaces to IT components and processes. SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains [196]. Taking a holistic and 'ecosystem' view, it deems such a context as a network of independent services, infrastructure and people who operate and affect those services [91]. Web Service is a direct implementation of SOA and offers facilities to enhance IT system communication where Web Services implements interfaces to corporate information system and processes are implemented as a "service chain" composed and orchestrated according to the needs. Such an architecture fits well the openness and flexibility required by Virtual Collaborative Organisation. Moreover, Cloud Computing based implementation provide a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [220]. Coupling SOA, Web Services and Cloud Computing models greatly improve interoperability, agility, flexibility and scalability among organizations.

As Collaborative business involves opening the partners Information System structure, outsourcing some functionalities to external provider, security is key factor while setting such inter-organizational coalitions. Whereas traditional IS security requirements and implementation focus on a static vision of information and processes, SOA implementation (which allows linking services together in arbitrary ways to meet user needs) requires a more dynamic approach, focusing on the way services and data may be used and allowing a dynamic implementation of security requirements, depending on the current context, user preferences and the way these constraints are propagated while composing and orchestrating services to address particular needs.

1.1 Context analysis

Different research and practitioners studies show that 'lost of control', 'data leakage' and 'contract breach' are the main concerns while moving to a federated business paradigm. A survey by IBM [189] shows that IT security (vulnerability to hackers and unauthorized access/use of company systems) is the leading concern of IT managers and CIOs. In this IBM survey, two thirds of respondents report that the investment upon collaborative paradigm such as Cloud Computing and SOA is considered risky. The two most serious risks are 'handling over sensitive data to a third party' and 'threat of data breach or loss'. Focusing on Cloud Computing, a Gartner's survey [143] reports the main risks as:

- Data location: Users must be convinced that their local privacy requirements are respected, even if they are not able to know the actual location of their data.
- Data segregation: The Cloud provider should prove its possession of encryption schemes, designed and tested by experienced specialists.
- Privileged user access: The Cloud Computing service providers must supply customers with specific information on hiring and oversight of privileged administrators.
- Regulatory Conformance: Providers must acquire third-party based audits and security certifications.
- Recovery: Providers should have the ability to achieve a complete restoration in due time after disaster.
- Investigative support: Investigating inappropriate or illegal activity is difficult in Cloud Computing, as log data may spread across an ever-changing set of hosts and data centers. However, it is essential that providers must support contractual commitment to investigation.
- Long-term viability: The data handled over the service side should be available after mass scale structural change, e.g. acquisition or even bankruptcy. Consequently, this report define as a "best practice" to 'ask potential providers how data will be get back and if it would be in a format that can fit a replacement application'.

In addition, a report by European Network and Information Security Agency (ENISA) [78] also studies the security benefits and risks brought by Cloud Computing. Their risk analysis detailed in four aspects of 'policy

and organizational risks', 'technical risks', 'legal risks' and 'general risks that have effects on Cloud Computing'. The report highlights several top security risks:

- Loss of governance: Clients cede some governance privilege to Cloud providers. This may affect security. At the same time the Service Level Agreements may not offer commitments to corresponding security service.
- Data protection: It may be difficult for the client to effectively check whether the data is handled in a lawful way. This problem is exacerbated in case of multiple data transfers.
- Compliance risks: Achieving certification (e.g., industry standard or regulatory requirements) may be risky as public Cloud infrastructure based implementation can even imply that some kinds of Conformance cannot be achieved.
- Insecure or incomplete data deletion: This issue presents higher risk for the client in the case of multi-tenancy and reuse of hardware resources.
- Malicious insider: Cloud architecture involves high risk roles, e.g., system administrators and security service providers.
- Isolation Failure: The multi-tenancy feature of Cloud Computing may lead to risks due to the lack of mechanisms that separates client data and privilege domains (e.g., guest-hopping attacks).
- Management interface compromise: Client access to provider through the internet leads to vulnerabilities related to remote access and web browser.
- Lock-in: With no standard and few tools support, it can be difficult for a client to migrate from one provider to another.

This report also recommends several research areas, namely, 'data protection in large scale cross-organizational system', 'building trust in the Cloud' and 'large scale computer systems engineering'.

Different "virtualization levels" can be used to set cloud-based IS implementation: one can use either Infrastructure as a Service models where only machines, networks and hardware resources are virtualized, Platform as a Service model which provides both hardware and solution stack virtualization (including DBMS, SOA support) or even Software as a Service, where the cloud provider hosts both the infrastructure and the application. Researchers from 'Institut National de Recherche en Informatique et en Au-

tomatique' (INRIA), and Hewlett-Packard (HP) [294] summarized threats related to the IaaS Cloud, namely, nefarious Cloud provider, malicious insiders, data loss and leakage, shared technology issues, interfaces weakness, account or service hijacking, unknown security profile (of companies leveraging Cloud service) due to complex infrastructure. It can be seen that the damage of these threats mostly relate to data misuse and leakage. The causes are generally due to the openness of business paradigm, complexity of IT infrastructure, and people factor.

Besides these studies, the increased adoption of web-based collaborative systems requires new research and developments. The surveys of Computer Science and Artificial Intelligence Lab in Massachusetts Institute of Technology [307] [77] [149] study the issues of privacy protection and copyright protection in Web-based collaborative system, e.g. Social Network Site and requirements of information accountability and fair use. 'Fair use' stands for the control of the information 'usage'. 'Accountability' means being able to track the derivation of information. This studies identifies several architectural factors according to these requirements.

- Provenance: It is necessary to track the information flow and record usage events at each endpoint so that each piece of information can be pinpointed down and its usage can be assessed against information provider's policy.
- Policy language framework: Policy defines which usage activities are appropriate or not. Coupled to provenance trail, this could support a "life-long" usage control. As far as collaborative context is concened, the heterogeneous endpoints involve being able to define a shared vocabulary to support interoperability requirements.
- Policy reasoning tool: System should include policy tools that not only apply policies over provenance to identify violations but also support reasoning request over policy, in a timing manner, to support runtime cooperation decision.

Lastly, other researchers have also analyzed the general characteristics of collaborative process [308]. Based on this and survey of traditional standalone Performance Measurements approaches, they propose a Performance Measurements approach for collaboration contexts. Six requirements upon VO members (partners) are identified: trust among the members; reliability; willingness to provide information and find solution; use of Information and

Communication Technologies; flexibility; promptness/speed. It can be seen that trust and reliability are important aspects.

The different requirements collected in both these surveys lead to a main conclusion: participants in collaborative information system take a holistic viewpoint upon security. They require an end-to-end security of their assets, asking not only for secured communication channel or partner-side security, but also for data privacy, due usage assurance and conformance mechanisms.

1.2 Related works

To fit these security requirements, the attention must be paid on the Information System implementation and deployment. In the Service Oriented Architecture (SOA) and Web Service (WS-) domain, several relevant standards and models have been proposed which discuss about security issues. For example the 'Reference Model for Service Oriented Architecture' [196] introduces Policy model and Contract model in brief. The 'Reference Architecture for Service Oriented Architecture' [91] discusses Governance and Security on conceptual level.

Securing web-services can be achieved at a technical level thanks to different standards used to integrate security "technical requirements" in the web services (from WS-Security to secure SOAP messages, to WS-federation used to define how different security realms can be federated, etc. [17] [226] [92] [14]). Nevertheless, these standards are mostly pertaining to the service 'secured delivery channel' and provide only an "instant" information protection (i.e. while processing a service or sending information to the service consumer). Despite the interest of this "instant" protection, the main problem in collaborative organisations is that once transferred on the consumer computing system, the information is no more under the service provider's control.

Paying attention on the deployment, strategies depend on the hosting infrastructure. In the Cloud Computing domain [323], several standards development organizations (SDO) as 'Organization for the Advancement of Structured Information Standards(OASIS)', 'Cloud Security Alliance (CSA)', 'European Network and Information Security Agency (ENISA)', 'Distributed Management Task Force (DMTF)', 'Open Grid Forum (OGF)', 'Storage Networking Industry Association (SNIA)' and 'Open Cloud Consortium (OCC)' are actively involved in the study and definition of the nature of Cloud Com-

puting. These organisations also collaborate to the development of Cloud Standards [230] [222] which are at their early stages. By now, only some guidance [73] are available. As Cloud Computing is a paradigm that uses different enabling technologies (e.g., Virtual Machine, API, network communication, Database, Web Service) and supports fine-grained, layered business models (i.e., SaaS, PaaS and IaaS) current propositions are mainly based on the security features of these corresponding technologies.

A major research challenged due to the collaboration paradigm is related to 'Trust assessment'. The basic idea is to 'let parties rate each other, for example after the completion of a transaction, and use the aggregated ratings about a given party to derive a trust or reputation score, which can assist other parties in deciding whether or not to transact with that party in the future' [147]. In this area, the focus is put on algorithms that aggregate peer estimations (from 'past' transactions), e.g., 'simple summation or average of ratings', 'Beta PDF and Bayesian system', 'Discrete Trust Model', 'Belief Model', 'Flow Model' [147]. Little attention is paid to the contractual regulation of consumers' actions upon assets (in 'current' or 'future' transaction).

In summary, the most developed aspects of security are 'point-to-point' security and 'trust' based security. The former aims at building up a secured environment for parties to communicate and cooperate, and does not take into account the way a digital asset is consumed. The later is rather an afterwards measure which focuses on scaling participants' security-related performance during the previous collaborations. Other researchers come up with proposals for end-to-end security in collaborative information system [16], but no detailed solution is available by now.

Digital Right Management (DRM) has been successfully introduced to protect music or video digital rights by providing a "playing" license that allows only a reduced use of the content. Coupling cryptographic techniques to adapted players, this solution provides an "end to end" protection to the contents and keeps them under the licensing control.

1.3 Contribution

Currently major IS security developments just offer limited protection for participants' assets value, restricted in 'point to point' security (secured service delivery), 'trust' assessment, etc. On the other hand DRM solutions can provide a life-long protection on some digital contents. Consequently, one can

take advantage of coupling these approaches to fit the security requirements involved by the collaborative context.

Adapting a DRM oriented organisation could fit corporate information policy requirements, providing adaptation to service composition and orchestration support. This leads to embed corporate security policy with the service both at the organizational and technological levels. This DRM adaptation to corporate IS context requires different extensions:

- First, the "policy languages" used by DRM systems, i.e. 'Rights expression languages' (RELs) [227] [305], don't have sufficient syntactic and semantic elements to incorporate security factors in a collaborative context. Namely factors from intra-organizational level and inter-organizational level are not taken into account. To overcome this limit, one should incorporate traditional security management methods and security metrics in collaborative corporate information system. To achieve this goal, we propose to extend the 'Rights Expression Language' (REL) to define 'usage actions' (in 'rights' element) and partners' attributes and attributes of the business process (as 'condition' elements).
- Second, the rights 'enforcement' mechanism of DRM can not be directly adopted in corporate IS context, as the assets consumption activities are carried out by consumer side processes (e.g. Web Service) which are out of the monitoring scope of traditional DRM monitoring module. To overcome this limit, one should extend security viewpoint to 'end-to-end' strategy by monitoring the 'usage actions' of asset consumer. To fit this goal, we propose a policy enforcement system that includes monitoring mechanism to check the consumer's conformance to asset providers' policies, as well as security mechanisms as encryption and digital signature to ensure the secured container for the asset and the secured delivery channel.
- Lastly, providers' benefits (intellectual property involved in services and information) should be maintained during service composition and information propagation. By now the DRM approach is not aware of this trait which is due to collaborative business process. To overcome this limit, one should protect assets in their full lifecycles, during the complex collaborative process. Propagating usage control is a complex operation as an asset can be 'merged' and 'converted' to other assets during the 'usage actions'. As an original asset serves as a part of the

derived asset, 'new use lifecycle', denoted as 'derived lifecycle' (and the lifecycle in which the original asset exists is denoted by 'initial lifecycle') must be managed to protect providers property. Therefore, merging assets involve aggregating their policies and potential conflicts should be detected.

As a summary, our research strategy relies on extending DRM with traditional security management and adapt it to fit the characteristics of collaborative corporate information systems, so that it can extend the assurance level to 'end-to-end' security.

An eligible solution will be a two-pronged problem:

- the expression of providers' requirements
- the policy negotiation (and policy aggregation when assets merge) process and the conformance of the agreements at 'implementation' level.

Our approach is organized with a 'top-down' strategy (see figure 1.1):

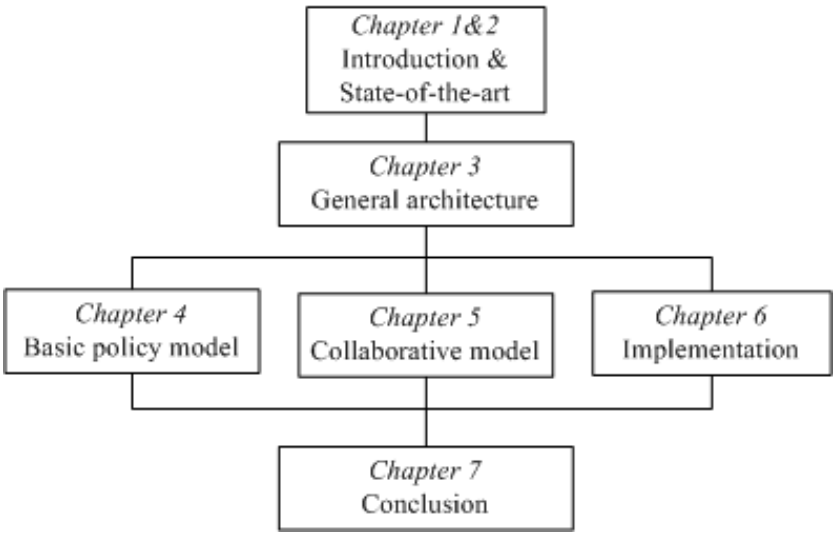


Figure 1.1: Thesis structure

- First, we consider the 'conceptual level' architecture to define which functionalities should be provided by our system (as functionalities used to manage partners policies, to inspect (and certify) partners security-related attributes, to mange the context, to monitor "usage activities").

- Second, we propose a policy model that incorporates the traditional security factors and 'usage control' factor as the foundation of 'end-to-end' security management. We extend the traditional DRM REL to build such a policy model, payinig attention on both the syntax and the semantics (to predict the effects of the policy evaluation process).
- Third, considering the traits of collaborative context, we define our policy model, using some rules (chosen by the policy owner) to extend the policy model to cover the whole business process (through policy aggregation). Besides, a method for deciding if policies should aggregate is necessary to adapt our policy model to specific collaborative contexts.
- Lastly, several components are necessary for supporting the policy negotiation (evaluation) and aggregation, the context management, the policy enforcement (usage monitoring) and the components for managing partners' policies, attributes and history activity records. We propose an implementation architecture aiming at providing comprehensive 'end-to-end' security management. We also develop several components and test their performances.

The enriched security information provided to participants can increase their willingness toward adopting SOA for their business. As for the Cloud Computing sector, our proposition can serve as a general framework for expressing participants' security concerns in the misc business models, e.g., DaaS, SaaS, PaaS, IaaS, etc.

Chapter 2

State of the art

Economic expectations lie in the flexible exchange of information system parts, which should allow a more differentiated coordination between organization and system design [290]. New paradigms for IT/Business process design are needed: Enterprise information system should be configured with the capability of adapting flexibly to operational requirements of shared process. This coordination in organization and system design can be implemented in SOA that promotes the 'agility' of companies and serves as a foundation for the implementation of a 'real-time enterprise' [245], paying attention to inter-operability and security.

Security is a topic related to many factors. Basically, security grounded on the infrastructure of Information System (IS). Factors in this point usually have fatal severity, e.g. physical level security of the devices; software level security (from Virtual machine, OS and libraries to applications), communication security, etc. Another basic aspect is the factor related to people. Security of this aspect rests with the issue of defining regulations on the usage and maintenance of corporate information assets, especially for coordinating users with different roles and different responsibilities. Other factors also have great impact on IS security. When coordinating multi-part interaction, either at intra-organization or inter-organization scale, identification is vital. This refers to authentication and authorization with certifications. Further, for inter-organization activity, factors as trust assessment, reputation are main considerations for cooperation. The emergence of collaborative computing systems, as Service Oriented Architecture (SOA), GRID and Cloud Computing, urged taking into account decentralized and collaborative paradigm impact. An important characteristic of these systems is that they

naturally support arbitrary, agile and complex interaction patterns. Information flow and asset value exchanges in these systems makes prevention of security leakage a tanglesome task. Another characteristic is that the autonomous ownership boundaries of the service and asset providers (or consumers) make the security configuration for the whole system even more arduous.

In this case, security can be achieved by focusing on the problem of: letting the right party access the right asset at the right time. That is, the party should have convenient security attributes, the asset should also have some security safeguard, and the access decision reflects the providers requirements. We naturally thought about using a comprehensive access control system as the basic model for the security configuration, provided we can incorporate related security factors in it. That is, basically, the miscellaneous security factors we considered can be settled down in this model. Then we want to ensure the End-to-End security. That is, being not only able to securely 'deliver' asset to consumer's access, but also to care about 'usage' upon it, in order to protect the provider's intellectual property. This is similar to Digital Right Management (DRM) in multimedia industry. End-to-End security is a natural generalization of DRM concept into corporate information system. Furthermore, with the trend of moving to collaborative computing like SOA and Cloud, the End-to-End security must cope with these new paradigms.

As a result, the whole system may seem to be complex and probably inconsistent. Nevertheless, it can be worthy to take advantage of existing building blocks, adopting them and making necessary adaption. This chapter introduces the state-of-the-art of security management in collaborative paradigm. It shows the awareness of security and the incompleteness of it. Lastly we expend the security perspectives from conventional intra-organizational security to global the general inter-organizational level. Then we introduce some technologies and issues relevant to support our goal of providing end-to-end security.

2.1 SOA: a new collaborative paradigm

The services ecosystem and other changes in inter-organizational business model make a heavy use of software services spanning multiple organizations. Such dynamic service chain system as well as the supply chain management system will provide agile support for business organizations. At the center

of such solution is the achievement of interoperability and security.

Interoperability relies on standardization and is bound to reduce human labor. It helps avoid information redundancy and inconsistency. Such standardization resides within both the architecture level and communication level design of business organization. By now, barriers are on one hand at an organizational level due to architectural governance in a business federation and, on the other hand, at the technical level, due to the technical information system implementation diversity. The Service Oriented Architecture allows an agile model for building both 'intra-' and 'inter-' organizational IS. The basic idea is not keeping application systems as stand-alone systems in enterprises, but rather 'pulling' the services needed from the Internet and then configuring them to fit the specific requirements [290].

2.1.1 SOA foundation

The SOA philosophy presents a new way to model IS federation in a collaborative context. The main feature of such a context is that instead of specifying an application hierarchy, it has to model the system as a network of peer-like entities with rules to control the interactions between participants.

In SOA, a service is a mechanism enabling access to one or more capabilities, using a prescribed interface and the execution constraints/policies specified by the service description [196], leading to de-coupled system, where exact implementation (technical detail) of a local service is not provided to the calling service, thus can be changed without impacting the calling service.

The basic elements of SOA model are (figure 2.1).

- Service provider is the party that provides resources and capabilities for specific needs as services. Service providers publish, unpublish and update their services. From a business perspective, the provider is the owner of the service. From an architectural perspective, this is the platform that holds the implementation of the service.
- Service requester is the participant who has a need that can be fulfilled by a service. From a business perspective, this is the business activity that requires a function to be fulfilled. From an architectural perspective, it's the application that is looking for a service and invokes it.
- Service broker, similar to a "yellow pages" service, is the party that provides a searchable repository of services descriptions, where service

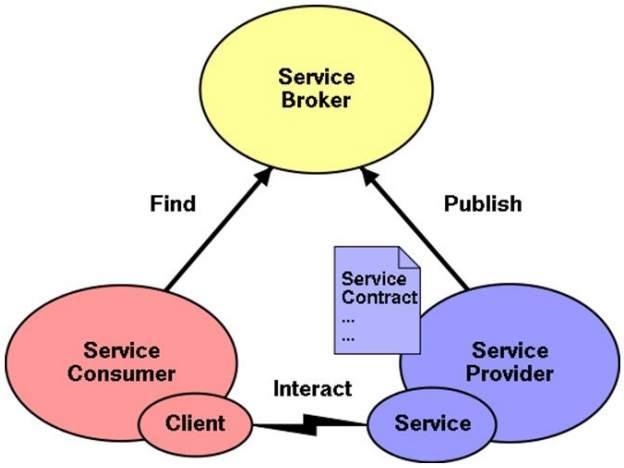


Figure 2.1: Basic Elements of SOA [126]

providers publish their services. Service requesters search and find the services they need through the broker and get binding information.

It is clear that since the service provider, the service broker and the service requester interact with each other, they should use standards for service description and communication. Thanks to this standardization, flexible and scalable complex applications are more easily achieved at either the intra-organizational or the inter-organizational level.

From a dynamic perspective, there are three fundamental concepts that are important for understanding what is involved while interacting with services: the visibility between service providers and consumers, the interaction between them, and the real world effect due to the interaction with a service.

Visibility refers to the capacity for those with needs and those with capabilities to be able to see each other. It introduces the possibilities for matching needs to capabilities (and vice versa). This is typically done by providing descriptions as functions and technical requirements, related constraints and policies, as well as access or response mechanisms. The descriptions need to use syntax and semantics widely accessible and understandable.

Interaction is the activity of using a capability. An interaction proceeds through a series of information exchanges and invoked actions, typically mediated by message exchanges. There are many facets of interaction; but they are all grounded in a particular execution context - the set of technical and business elements that form a path between the needs and the capabilities.

This allows service providers and consumers to interact. It also provided a decision point for any policies and contracts.

The purpose of using a capability consists in having one or more real world effects. An interaction is "an act" as opposed to "an object" and the result of an interaction is an effect (or a set/series of effects), as returning information or changing the state of entities that are involved in the interaction.

2.1.2 SOA impacts on the organization

SOA provides a new way of modeling social and business federation structure. The business mode in a SOA-based system is characterized in terms of providing services and consuming services to achieve mutually desirable real world effects [91].

In the 'business via services' view, a service is the mechanism by which needs and capabilities are brought together. A Stakeholder is an individual entity (human or non-human), or an organization of entities that share interest in services and/or the outcomes of service interactions. A participant is a stakeholder that has the capability to act in the context of a SOA-based system. A service provider is a participant that offers a service that permits some capability to be used by other participants. A service consumer is a participant that interacts with a service in order to access a capability to fulfill a need. A service mediator is a participant that facilitates the offering or use of services in some way. An agent is any entity that is able to act on behalf of a person or organization. There are two main classes of non-participatory stakeholders: third parties, who are affected by someone's use or provisioning of a service, and regulatory agencies who wish to control the outcome of service interactions.

In this view, a stakeholder has ownership over resources (e.g. capabilities, information), shares them beyond ownership boundaries, fulfills consumers' needs with real world effects (the result of service action) and benefits from the service and resources that he offers out.

SOA is a mean of organizing solutions that promotes reuse, growth and interoperability. It is not itself a particular solution to domain problems but rather an organization and delivery paradigm that enables to get more value from use both of capabilities which are locally "owned" and those under the control of others. It also enables one to provide solutions in a way that can be easily modified or alternated.

SOA is scalable and offers agility to business process modeling

'The main value of SOA is that it provides a simple scalable paradigm for organizing large networks of systems that require interoperability to realize the value inherent in the individual components. ' [196]

Services can be dynamically composed. The composition of services is the act of aggregating or "composing" a single service from one or more other services. Two types of services can be identified: Atomic Service and Composite Service. Both of them are visible to a service consumer (or agent) via a single interface and are described via a single service description. The composite service consists in the aggregation of atomic services or other composite services.

In SOA, business processes are defined as a set of consistent actions implemented by services. Performed in a logical sequence over a period of time and with appropriate rules applied, this service chain provides a business outcome. A "Service-oriented business process" means that the aggregation or composition of all of the abstracted activities, flows, and rules that govern a business process can themselves be abstracted as services.

Besides, between trading partners that span organizational boundaries often occurs the business federations supporting a "peer"-style interactions: partners act as equals without a central coordination.

When services are used as encapsulations of capabilities (with an arbitrary granularity) more dynamic organization can be set: it is possible to let in or remove participants and stakeholders according to the business requirements.

SOA provides a scalable environment as it makes the fewest possible assumptions about the network and minimizes trust assumptions that are often implicitly made in smaller scale systems. To develop systems that are scalable, evolving and manageable, an architect using SOA principles is better equipped. It is also easier to decide how to integrate functionalities across ownership boundaries. The IT infrastructure based on SOA is also more agile and responsive than one built on an exponential number of pair-wise interfaces. For example, a large company that acquires a smaller company must determine how to integrate the acquired company's IT infrastructure into its overall IT portfolio. Through its inherent ability to scale and evolve, SOA enables an IT portfolio which is also adaptable to the various needs of a specific problem domain or process architecture. It inherently supports the corporate information system aggregation.

SOA supports contracts based on policies and enables fine grained virtual organization using business processes of participants. Integrating people relationships in collaborative business process supported by

SOA involves being able to set commitments and enforce them according to a community strategy. It includes several key elements.

An agreement shared by a group of participants defines a social structure. A Role is an identified relationship between a participant and a social structure that defines the rights, responsibilities, qualifications, and authorities of a participant within the social structure context. Right is a predetermined permission that permits an agent, i.e. any entity that is capable of acting on behalf of a person or organization, to perform some actions or adopt a stance in relation to the social structure and other agents. Authority refers to the right to act as an agent on behalf of an organization or another person. Responsibility is an obligation associated to a role player who has to perform some actions or to adopt a stance in relation to other role players.

2.1.3 Policy

A policy is defined by a set of assertions. The SOA reference architecture [91] provides mechanisms to enforce policies and contracts to support automated governance and ensure efficient operations in a consistent way with the goals of the social structure.

Assertions and commitments are defined as propositions - an expression of some property of the world whose truth can be measured by examining the world and checking that the expression and the world are consistent with each other. Assertions are claims about current state while commitments are agreements to future state.

The SOA reference architecture identifies two types of constraint mechanisms [91]. The permission-style constraint defines the right to access some resources or to perform some actions. The obligation-style constraint defines the requirement to perform some actions or maintain the state of a resource. This architecture also identifies the key components of the constraint mechanisms [91]:

- Policy/Contract administration point, for allowing participants to manage policies;
- Policy Distribution/Repository, storing policies to be used by Decision Points;
- Attribute Information Point, collecting and forwarding attributes (Named values that define characteristics of participants, resources, actions, or the environment) to the Decision Point;

- Decision Point, evaluating participant requests against relevant policies/contracts and attributes to render a permission decision;
- Enforcement Point, enforcing and assuring the Decision Point decisions and obligations;
- Measurement Point, Identifying mechanisms for measuring and monitoring policy obligations;
- Audit Point, recording participant actions and measuring results of obligations.

With emphasis on governance, policy and contract management, SOA allows the stakeholders to negotiate and set key policies that govern the system at runtime. By this way, the SOA reference architecture provides a framework for characterizing the conditions and obligations of service based business federation process.

2.1.4 SOA implementation

A Web service is defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network" [125].

The basic activities of Web Service life cycle include creation, description, publication, discovery, invocation and un-publication associated to the 'basic layer', as shown in figure 2.2.

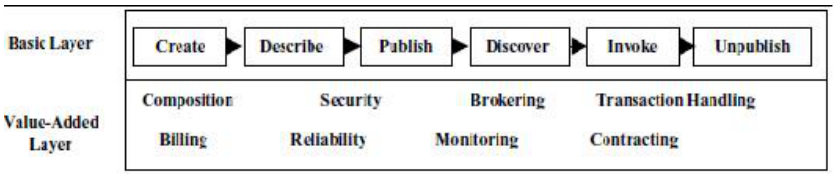


Figure 2.2: The Web Service lifecycle [292]

The 'value-added layer' covers the activities that bring better performance to Web Service environment, such as composition, security, brokering, reliability, billing, monitoring, transaction handling and contracting (see figure 2.3).

The core layer is devoted to communication via network, using mostly SOAP and XML for message (and data) exchanges. Other standardized mechanisms are also used to increase performance, as WS-routing, WS-addressing [42] and WS-reliability [92].

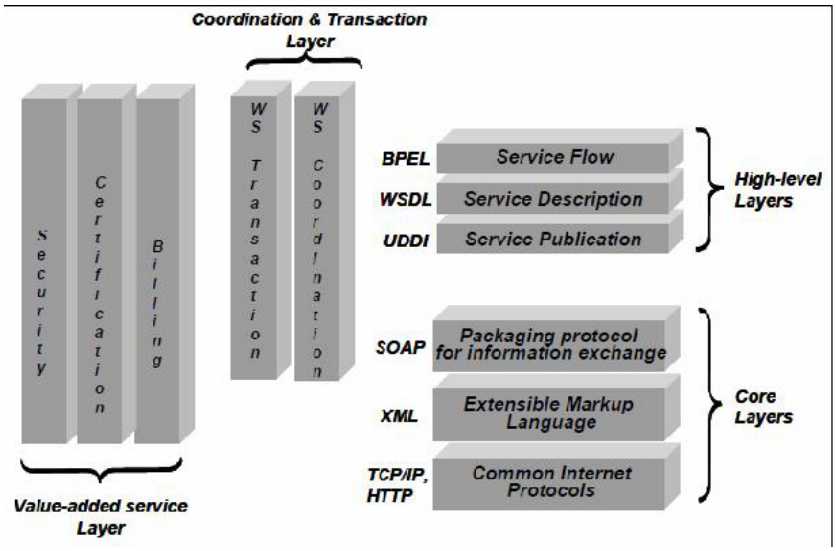


Figure 2.3: The Web Services technology stack [236]

High-level layers are used for Web Service description, publication, selection and coordination. XML-based representation is the predominant mechanism: WSDL [58] [57], xCBL, cXML, UBL [41] and ebXML [87]. Research also develops Ontology-based representation: OWL [202], WSMF [93],WSML [80] AND WSMO [79]. UDDI(Universal Description Discovery and Integration) [223] is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), enabling businesses to publish services, discover each other and define how the services or software applications interact over the Internet. Other mechanisms for this task are semantic UDDI [231] and ebXML [87]. Web Service Coordination, orchestration and choreography are issues concerning the management of Web Services cooperation. For such tasks, WS-BPSS (from the BPSS [87]) and WS-BPEL [145] are the main standards (compared to WS-CDL [158]). Such technologies allow dynamic governance of complex cooperation pattern among multiple service providers and consumers.

These mechanisms are basic building blocks for bridging the gap between the conceptual definition of SOA and the business federation environments. In addition, there are other important issues that need to be considered:

- **Monitoring**
  It covers three aspects. 'Transaction' defines how to 'undo' the relative

WS's task when a WS failed, borrowing the ACID (atomicity, consistency, durability and isolation) concept to WS. 'Change management' concerns administrating quitted and newly joined Services. 'Optimization' chooses 'best' Service among several ones according to their Qos features (which is used both in the orchestration stages and in the change management).

- **Qos**
  Major requirements [236] [317] include runtime properties as availability, accessibility, conformance to standards, integrity, performance, reliability, scalability, etc. and business qualities as cost, reputation, regulatory, etc.
- **Privacy and Security**
  Security (especially application level security) [317] includes authentication, authorization, non-repudiation (keeping historical executing data), message integrity and confidentiality, operational defence, etc. Standards as SAML (security assertion markup language) [226] facilitate security across enterprises. Privacy issues have been taken into consideration in the W3C's P3P [69] and privacy policies in OWL-S [148].
- **Policy**
  Policy represents a set of specifications that describes the capabilities and constraints on security, Quality of Service or other aspects. W3C has proposed a recommendation of WS-Policy in September 2007, which is a specification documentation that allows both web services to use XML to advertise their policies and web service consumers to specify their policy requirements.
- **Interoperability**
  The 'basic profile1.0' by WS-I (Web Services Interoperability Organization) is the baseline for Web Service interoperability. As SOA is a decentralized and collaborative architecture, interoperability is the fundamental ability to support other features. Many efforts are being made in research and practices to enhance interoperability in both the syntactic and semantic levels. For instance, the XML provides a standardized and structured data organization format and is more and more employed for exchanging messages. On the other hand, semantic technologies are increasingly used for representing Web Service's features and for matching Web Services [28] [148] [199] [231] [133].

Above Web Service features indicate a gap between the requirements of SOA and the existing IT infrastructure. Responses for these requirements rely on re-organizing or upgrading current information systems, making new technologies adapting to existing ones and achieving features necessary for collaborative business process. The ESB [243] solution is a influential approach for this purpose.

To implement an information system in SOA, a highly distributed communications and integration backbone is required. This functionality is provided by the Enterprise Service Bus (ESB) that is an integration platform that uses Web Services standards to support a large variety of communication patterns over multiple transport protocols and deliver value-added capabilities for SOA applications [234]. It supports capabilities such as service orchestration, intelligent routing, provisioning, and service management [234] and also guarantees the security of data and services. The extended SOA (so-called 'xSOA') [233] [235](see figure 2.4) addresses such requirements.

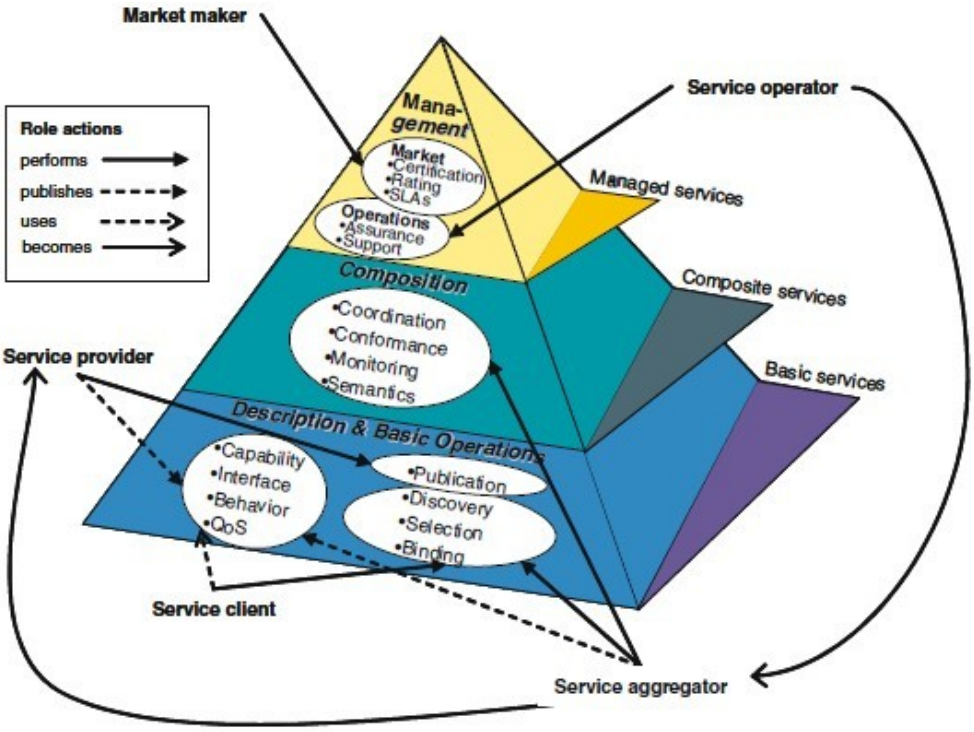


Figure 2.4: Extending SOA [234]

The xSOA is a multi-level architecture which embraces a multi-dimensional separation of concerns based on the need to separate basic service capabilities from the functionalities for service composition and management. In such architecture each layer uses the functionalities provided by the lower-level components predecessor layer to accomplish its task.

The ESB middleware provides services at different layers: communication, routing, translation and discovery are basic services whereas the composition and management services are associated to the composite service layer, lastly, SLA and security are used to set managed services.

Consequently, an ESB is an integration platform of middleware based on de facto standards that provides basic functionalities such as message-based exchange, data transformation and intelligent routing in a highly distributed architecture via an event-driven and standards-based messaging engine [279] (see figure 2.5).

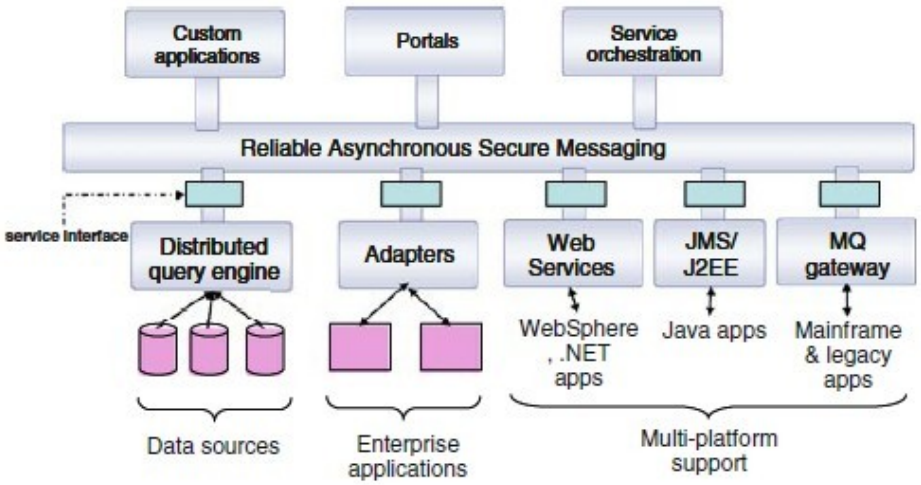


Figure 2.5: ESB connecting diverse applications and technologies [234]

ESB provides the distributed processing, standards-based integration and enterprise-class backbone required by the extended enterprise [200]. In the enterprise context, business events (e.g. a customer order, the arrival of a shipment at a loading dock or the payment of a bill) may affect the normal course of a business process at any time [234]. This implies that business processes cannot be designed a priori assuming that events are predetermined and follows a particular flow: they must be defined dynamically, driven by

incoming, parallel and a synchronous event flows [203]. Business processes can be implemented using an event-driven SOA [282] [278], based on service contracts and other associated meta-data such as policies.

SOA provides a uniform mean to offer, discover, interact with and use capabilities in an environment that transcends ownership domains. This brings into concern the ownership and other issues related to SOA governance. SOA governance applies to three aspects of service:

- SOA infrastructure - the "plumbing" that provides utility functions that enable and support the use of the service;
- Service inventory - the requirements on a service to allow it being accessed within the infrastructure;
- Participant interaction - the consistent expectations according to which all participants are expected to comply (well behaved).

SOA governance should provide an end-to-end protection of assets benefitting provider's ownership rights during the full business federation process lifecycle. This leads to establishing contracts among providers and consumers to control the behavior in the run-time space, paying attention on both service and infrastructure layer (e.g. is the used system corrupted or not). An example is the 'PEtALS Master' SOA Governance solution [244] (see figure 2.6). Built as an upper layer on the ESB, it offers components for service registration, information repository and SLA (contract) [35] management.

At the deployment stage, attention must be paid on the distributed system organization. For example in the opensource ESB PEtALS, different ESB nodes can be federated to support business federation of an expended scale. This requires the adaptation to distributed architecture when proposing an end-to-end security management solution.

Such distributed architecture can implement on 'virtualized' IS infrastructure. This requires a security management method coping with layered IS infrastructure. A typical application based on layered IS infrastructure is Cloud Computing.

2.1.5 Cloud Computing with Web Service

Cloud Computing leverages Web Service technology for the provisioning and delivery of service. It diminishes the overhead of pre-planning for provisioning. Thus enterprises can acquire services from cloud providers starting

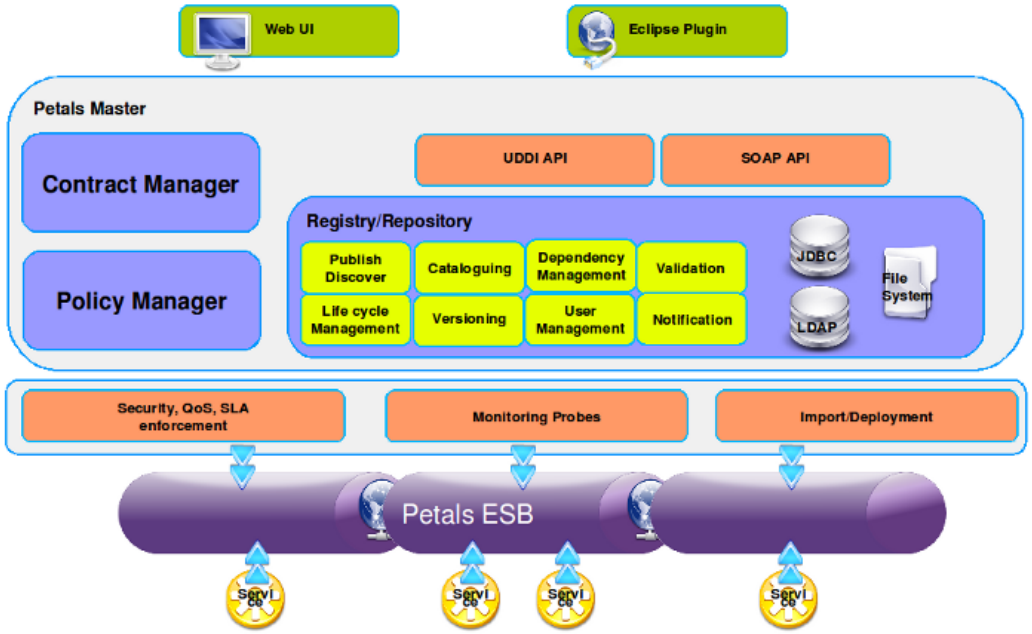


Figure 2.6: PEtALS infrastructure [244]

from small scale and increase investment according to business development. Cloud Computing supports a layered service-driven business model [323] (see figure 2.7):

- **Infrastructure as a Service(IaaS)**
  It stands for on-demand provisioning of infrastructure, mostly via Virtual Machines. Examples include Amazon EC2 [12], GoGrid [64] and Flexiscale [99].
- **Platform as a Service(PaaS)**
  It stands for providing platform layer resources, namely operating system support and software development frameworks. Examples are Google App Engine [116], Microsoft Windows Azure [204] and Force.com [261].
- **Software as a Service(SaaS)**
  It stands for providing on-demand applications over the Internet. Examples include Salesforce.com [261], Rackspace [251] and SAP Business ByDesign [266].

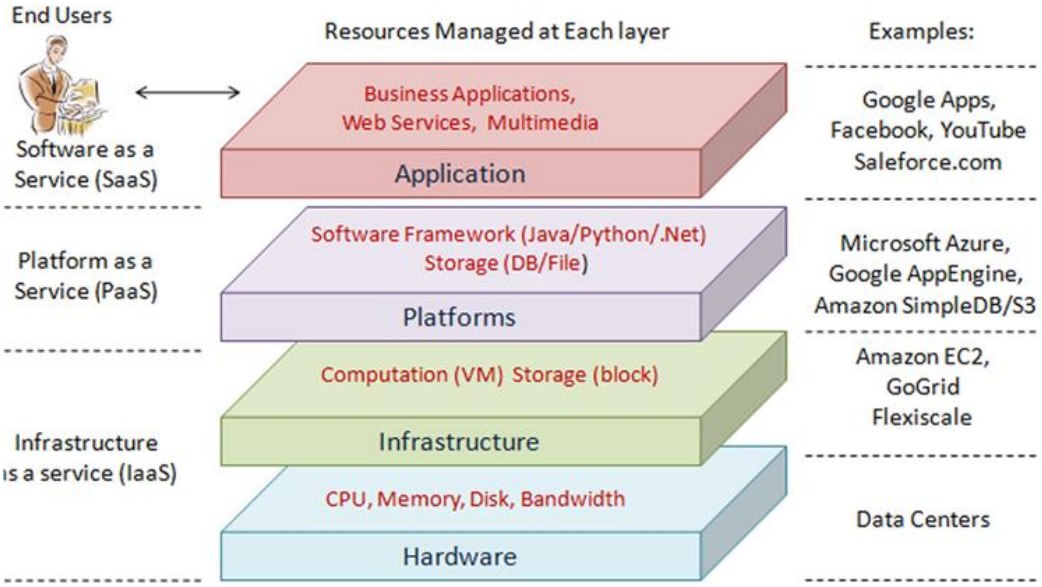


Figure 2.7: Cloud computing architecture [323]

Such a layer scheme resembles the Software Stack viewpoint upon information system. The patrimonial value of a Cloud provider involves the hardware and software value of corporate information system and organizational intellectual property. A Cloud owner exhibits the patrimonial value as Cloud service and requires protecting its interests. Upper layer Cloud provider relies on the services of lower layers which can be offered by other cloud owners. Security governance of collaborative context as a Cloud Computing context should be able to identify the security factors in the different layers and manage them separately in a "security stack" organisation.

Security issues at each Software Stack persist in the corresponding Cloud service layer. In addition, the providers' security profiles must be consistent to meet the consumers' security requirements. Service level agreement (SLA) are built among Cloud providers and consumers to accommodates their requirements and protect corporate patrimony. These SLAs have dedicated part for security management and intellectual property protection. This design requires an adapted risk analysis and management method, fitting the cloud vision, paying attention on security specification and on the way it is implemented.

2.2 Security foundation of Information System

Inter-enterprise business process engineering specification has to satisfy two goals: respecting each enterprise autonomy and patrimony while building a consistent dynamic and multiple super-enterprise organization. This involves setting a "common" information system with well defined processes able to support efficiently both formal and informal collaboration [37], interconnecting the different partners' own information systems and respecting each partner's security strategy. Consequently, key points are risk analysis and security management.

2.2.1 Corporate IS risk analysis and management

Since the 1980s, several methods and standards have been developed to identify risks and address security issues for corporate Information System (IS). Using these methods, one can identify and design the security features of the corporate IS. Such features can serve as elementary criteria to consider the inter-organizational cooperation decision.

EBIOS [83] (Expression des Besoins et Identification des Objectifs de Securité) has been created by the DCSSI (Direction Centrale de la Sécurité Des Systèmes d'Information), a department of the French Ministry of Defence. It provides a method and comprehensive information (e.g. detailed descriptions, strategic stakes, detailed risks with their impact on the organization, explicit security objectives and requirements) to support decision-making regarding the security policy. This method is an exhaustive approach and gives a greater awareness for everyone involved in a project. Unlike scenario-based risk analysis approaches, this method has great generality for many application domains. Consequently it can be used in different contexts.

ISO/IEC 17799 [140] and ISO/IEC 27002 [141] offer guidelines for information security management. The standard contains twelve main sections concerning: risk assessment, security policy, human resources security, physical and environmental security, communications and operations management, security implication, conformance ensuring, incident management. It consists in a set of policies, standards, procedures, and guidelines that well match most of applications. They also provide certifications. This allows organizations to show their own security level as well as comparing it with the

security features of their trading partners and competitors.

OCTAVE [11] is a self-directed, risk-based strategic assessment and planning technic for security. Unlike the typical technology-focused assessment methods, OCTAVE is targeted at organizational risk and focused on strategic, practice-related issues. OCTAVE is also an asset-driven evaluation approach. Its process is led by identifying and rating the assets in the corporate IS, analyzing the risks and their severities, collecting security requirements and proposing security policies.

SNA [90] describes a process for systematically refining an enterprise system architecture to resist, recognize, and recover from deliberate, malicious attacks by applying reusable design primitives. It is an iterative risk-driven process which adopts the Spiral Model [38].

MEHARI [65] is a complete method of evaluation and management of risks associated with information, treatments and resources used, provided by the CLUSIF (Club de la Sécurité de l'Information Français). It is built around a comprehensive set of modules, tools and questionnaires and is distributed under the Open Source principles (downloaded and applied in over 100 countries). MEHARI knowledge base provides methodological framework, tools and documentation for guiding security management tasks as follows:

- The '**major stakes analysis**' focuses on the objectives and expectations of the organization's business units. It provides a scale of harm value resulting from security incidents and a formal classification of 'primary assets' (processes, information) and 'supporting assets' (including premises, offices, IT and networks, etc.).
- The '**analysis of the vulnerabilities**' focuses on the effectiveness of the security services, their firmness and their permanency over time. Its consideration includes both the information system and the work flow environment.
- The '**decreasing and managing the risks**' task provides, generally speaking, assessments of the intrinsic level of consequences of the risk situation and an optimal setting of action plans to reduce the risk.
- The '**monitor of the information security**' step uses several 'indicators' to compare the results of action plans to the objectives. It provides synthetic reports about: risk and vulnerability levels, security themes (16 criteria such as access control, continuity planning, etc.), compliance measurement to all ISO 17799:2005 controls and dashboard

of critical risks.

These methods and standards are recognized as most successful approaches for corporate IS security. Their implementations allow collecting security issues in a business federation from the very fundamental aspects. The analysis drills down through both the 'inter' and 'intra' organization level.

2.2.2 Intra-organizational security

Security issues at this level can be organized throughout the 'software stack' layer. Software Stack in its general sense stands for the layered partition of all the resources requested by a computing task, namely the hardware layer, the virtualization layer, the operating system layer, the middleware layer, the network layer, the application layer and the people/organization layer. Generally, an uper-layer relies on the functionalities of the under-layer, including the 'trustworthiness' of it in a security point of view [54]. Enterprise security policy takes a holistic viewpoint where each layer has different security goals and management aspects.

The **hardware layer** comprises management of the physical resources, i.e. servers, routers, switches, as well as electrical supply and cooling systems. The physical security is vital to the integrity, confidentiality and availability of enterprise information system. This point is emphasized by many security management methods and standards as EBIOS [83], OCTAVE [11] and ISO/IEC 17799/27002 [140]. Typical issues involve hardware configuration, fault tolerance, power and cooling resource management, physical access control, etc.

The virtualization layer maps the physical resources to a pool of 'sliced' and 're-organized' storage and computing resources using virtualization technologies such as Xen [311], KVM [171] and VMware [296]. Due to the adoption of Cloud Computing to support applications, the virtual machine security is drawing industry and research concerns [294] [84] [144] [211]. Virtual machine security is a new fast-growing field (we should follow the progress and include new threat and security mechanism into our solution). One needs to consider the protection of both the guest OS (VM) and the hypervisor (VMM). VM security requires isolation and proper management of interaction between VM and hypervisor. For example cryptographic protection and access control can protect VM image at the storage and transportation stages. Encrypted boot and data partition protect VM at the

deployment stage. VM introspection monitors the deviation from 'normal' behavior at runtime. Network traffic authentication can be used to prevent guest-to-guest monitoring attack. Hypervisor protection aims at developing counter-measures to face the main threat: VM escape, where the guest OS may exploit the security flow of VMM, elevate its privilege and compromise the host environment. This problem can be solved by properly configuring the environment, timely path update and audit trail. In addition, there are also threats due to external modification of VM or hypervisor. The solution generally leverages digital signature technology, e.g. the trusted bootstrap.

The **operating system** security underwent a relative long history. Some works [163] [163] summarized and categorized security issues at this layer. Confidentiality, integrity and availability stakes are mostly challenging at the operating system level. Many mechanisms have been developed, as digital signature algorithms, encryption algorithms, key exchange algorithms and checksum algorithms, to enhance authentication, whereas Access Control and credential management ensure authorization. The recent years have seen trusted Platform Module(TPM) and trusted-boot more and more adopted to face rootkit attacks.

The middleware layer offers libraries and supports functionalities to build an upper-layer environment above operating systems to facilitate applications deployment. Examples include Java virtual machine, .NET runtime, Google App Engine and Windows Azure. This layer is vulnerable to attacks as SQL injection, etc. Counter measures as continuous path updating and version management can mitigate such vulnerabilities. Encryption and signature can also be used in data storage service.

The **network layer** is associated to the communication mechanisms (network and messaging) between hosts to support service delivery. Table 2.1 shows a list of security mechanisms in TCP/IP network.

	Security requirements	**Fulfilling Technologies**
6	Transport level audit trail	Logging, audit trail policy
5	Transport level accessibility	Firewall, IDS, IPS
4	Transport level user access control	TLS, SSL, IPSec
3	Transport level user authentication	TLS, SSL, IPSec
2	Transport level data integrity	TLS, SSL, IPSec
1	Transport level data confidentiality	TLS, SSL, IPSec

Table 2.1: Network Security Level (TCP/IP security) Stack

Security protocol as 'Internet Protocol Security' (IPsec), 'Secure Sockets Layer' (SSL), and 'Transport Layer Security' (TLS) provide security services as confidentiality, integrity and authentication. They also facilitate access control. Firewall, Intrusion Detection System and Intrusion Prevention System enable more comprehensive mitigation of network level threats. Logging and audit trail offer 'non-repudation' thus enhancing security level in network layer.

The **application layer** consists in utilities to perform specific computing tasks e.g. some SaaS application as Google Apps, Facebook, Youtube, etc. Threats are more related to applications and commercial models. A major challenge is the protection of enterprise privacy, e.g. the infringement of copyright, leakage of digital content, business secret exploition, etc. This can be caused by mal-behavior of employees, clients or providers, abuse of account, or failure of the underlying layers. As application data bears the intellectual property, unauthorized usage of them is a major risk that hinders the cooperation motivation. Access control and encryption can mitigate the risk. Auditing trail is critical for forensic evidence.

The **organization and people layer** refers to the organizational regulations to enhance security, namely security management strategy. Such regulation usually defines the privilege and obligation of people, their positions, requirements and maintenance rules for devices, utilities and environment aspect. In fact, management issue is a global scale topic and spreads in every layer. We extract them to emphasize their impact and introduce some wide adopted approaches.

Based on the stack viewpoint, IS security can be scaled in a structural manner that adapts to the collaborative computing contexts as Cloud Computing, GRID and SOA. Categorization of security factors based on the layered analysis can serve as a part of the 'vocabulary' for expressing enterprise security level.

Nonetheless, these factors reflect only the centralized security configuration within the scope of one organization. A global security level of business federation can not be concluded without an inter-organizational security analysis.

2.2.3 Inter-organizational security

Recent years have seen the development of the new collaborative paradigm where services are organized dynamically in different service-chains to sup-

port various business processes. It involves re-thinking the security policy at an inter-organizational level. Being autonomic entity, each party has its own criteria and policy about when and how it will share services and/or values with others. In such context, trust between partners plays a critical role to smooth cooperation and information sharing across different trust domains [310]. It is a driving force to set collaborative organization. In short, trust can be defined as an assertion on the behaviors of participants in relation to each other [91].

An identification service is the foundation for trust relationship. Centralized or decentralized authentication authority is required to identify partners. A decentralized trust authority provides individual participants more autonomy to authenticate and authorize actions and events to support collaboration. Nevertheless this involves building a trust-chain. A common trusted third party may be used to facilitate trust chain enactment. An example of this strategy is the policy-based exchange of certificate based on PKI service.

Trustworthiness of a partner can be computed by direct-trust, recommendation and reputation (see table 2.2).

Trust level	**Trust type**	**Trust source**
Direct-trust	Direct-trust	Direct assessment
Indirect-trust	Recommendation	Recommendation by authority
Indirect-trust	Recommendation	Recommendation by peers
Indirect-trust	Reputation	Convention-based
Indirect-trust	Reputation	Authority
Indirect-trust	Reputation	Peer-opinion

Table 2.2: Trust assessment

Direct-trust is a model where a party estimates the trustworthiness of a partner according to their interaction history. Usually, a party scales the partner by factors like the number of successful interactions, frequency, elapsed time from last interaction, etc.

Indirect-trust means that a party estimates the trustworthiness of a partner based on opinions of other partners. There are two types of indirect-trust assessment tasks, namely recommendation and reputation. Recommendation concerns generating an aggregated opinion, dealing with the more 'popular' partners, e.g. those with better QoS, more visiting rate, etc. The reputation model [123] is rather used to deal with security issues. It generates scales

upon security attributes of partners and records them. Reputation records can be generated from three sources. Convention-based model uses standards, e.g. ISO/IEC27002 [141], to regulate and certificate party's quality. Authority model is used in e-commerce where some organizations grant certificates (trust seals) for proof of the QoS/security of websites. Peer-opinion model receives most research attentions and practice adoption. Algorithms to aggregate peer estimations can be classified into 8 categories: simple summation or average of ratings, Beta PDF and Bayesian system, Discrete Trust Model, Belief Model, Flow Model [147], Fuzzy Model where membership functions are associated to trustworthy, Hidden Markov Models [206], which takes the time between observations into account, and Entropy [197] based models.

Access control can be seen as a fine-grained trust model where partners' properties are evaluated to decide the access to assets. The fast evolution of Web-based distributed computing paradigm has risen a variety of resource protection requirements. The work done by the security research community to address these requirements has led to the definition of a number of access control models [8], e.g. DAC (discretionary access control), MAC (mandatory access control) and RBAC [263] [94].

Role-based Access Control (RBAC) is a security model which relies on the organizational view to provide authorization to access resource. The RBAC model creates an indirect relationship between rights and actors through the roles played by the actors. Thus, security policies are defined to govern roles instead of individual users which facilitate the management of security policies [279]. RBAC 'has emerged as a full-fledged model as mature as conventional mandatory access control (MAC) and discretionary access control (DAC) concepts' [95]. It has wined extensive real-world adoption thanks to its features as perspicuity of role administration and supporting of Static Separation of Duty (SSOD) and Dynamic Separation of Duty (DSOD). The Role-Based Access Control (RBAC) model often uses a manual assignment of users to appropriate roles. When the service-providing enterprise has a massive customer base, assigning users to roles ought to be automated [9]. Open environments such as Internet involves that service requesters are not only defined by an identifier but by a set of attributes (usually substantiated by certificates) to gain accesses to resources [303]. Owing to its centralized administration model, RBAC does not adapt naturally to the decentralized collaborative ecosystem paradigm.

Organization Based Access Control (OrBAC) [152] is a policy model in-

cluding permissions, prohibitions and obligations. It can be used to define policies across the organizational boundaries. In this model, each security policy is defined for and by an organization. By this way it is possible to handle simultaneously several security policies associated with different organizations.

The recent past has seen many attribute-based access control systems [39] [40] [318] [319] [320]. The Attribute-based access control model is a decentralized model and is supported naturally by XACML model. Past actions and Reputation records can be modeled as subject attribute in such model. It is able to express RBAC by leveraging semantic [98].These features make attribute-based access control an adaptive policy model for trust-based access control.

Building trust relationships usually relies on a Service Level Agreement (SLA), paying attention to security aspect. Such agreement consists in a policy part regulating the partners' privilege in order to protect each other's benefits. Some mechanisms for monitoring party's behavior can be set to determine the effectiveness of such agreement. Monitoring mechanisms vary according to the implementation context and technical details. At the network level, network traffic and message exchange analysis between service provider and consumer, e.g. based on SOAP message intercepting [209], can be used. The consumer behavior can also be monitored at the system level by observing execution log, e.g. hooking, system call logging and Runtime verification [22].

In this section we introduced briefly the state-of-the-art of security management approaches and achievements, at both intra-organizational and inter-organizational level, to set comprehensive security and assurance system for collaborative enterprise information system. Nevertheless, these factors provide an "instant point of view", i.e. the partner selection, the service secure delivery, countermeasures to external attacks. A comprehensive security and assurance system for collaborative enterprise information system should not be confined within these factors, as a federated organization usually requires a life-cycle long protection for services and information.

2.2.4 Security and governance in SOA and Web Service

In SOA, security can be identified as the confidence that services are enhanced to prevent accidental or malign intent of other people to damage or compromise trust in the system [196] [91]. Governance is used to align decisions with the overall organizational strategy and enterprise culture.

SOA characterizes the key security concepts [141] as follows:

- Confidentiality refers to the assurance that unauthorized entities are not able to read messages or parts of messages that are transmitted.
- Integrity refers to the assurance that information that has been exchanged has not been altered.
- Availability concerns the ability of systems to use and offer the services for which they were designed.
- Authentication concerns the proof of a participant's identity.
- Authorization ensures that the information and actions that are exchanged are either explicitly or implicitly approved.
- Non-repudiation concerns the accountability of participants: they are not able to later deny their actions.

Threats can come from either third parties from outside or from participants in the system. The later is of particular concern as SOA system itself is an ecosystem spanning multiple ownership boundaries.

The SOA reference architecture [91] lists common threat types:

- Message alteration: the attacker modifies the content of message.
- Message interception: the attacker intercepts and understands message between participants.
- Man in the middle: the attacker intervenes in the conversation and convinces each participant that he is their real correspondent.
- Spoofing: the attacker convinces a participant that he is someone that the participant should trust.
- Denial of service: the attacker prevents legitimate users from making use of the service.
- Replay: the attacker captures the message traffic during a legitimate interaction and then replays part of it to the target and persuades it to respond as though it was a legitimate interaction.

- False reputation: a malicious user completes a normal transaction and then later attempts to deny that the transaction occurred.

Performing threat assessments, devising mitigation strategies and determining acceptable levels of risk are the foundation for an effective process to mitigate threats in a cost-effective way. It can include:

- Privacy Enforcement, which is usually maintained by encryption and 'Policy Decision Points (PDP)' / 'Policy Enforcement Points (PEP)'. A 'PEP' for enforcing privacy can be an automatic function to encrypt messages as they leave a trusted boundary or simply ensuring that such messages have been suitably encrypted.
- Integrity protection, which uses digital signature to provide protection against message tampering or inadvertent message alteration.
- Message Replay Protection, which controls the message 'unicity' by using a message ID, a timestamp or seed information which can be used by the reply message.
- Auditing and logging, which are functions that maintain careful and complete logs of interactions. They can be used for auditing purposes.

Web services security quality is the ability to determinate the legality of access to the system and service, providing integrated security service for the use of stable, reliable and appropriate authority in order to reduce or eliminate all potential threats, which may occur while using Web services [225]. WS standards propose message level mechanisms for SOAP and XML security, offering enhanced security (see table 2.3).

SOA security includes not only the security services as integrity, availability, accessibility, reliability, and confidentiality etc: it emphasizes on using policy as a further means for security governance and management. Nevertheless, the WS standards address mainly the issue of 'secure delivery channel', with falls in the network layer of the software stack model. Thus a more comprehensive solution is expected.

2.2.5 End-to-end security

In a global security perspective, the IS architecture for business federation is an ecosystem where all the organizations act as independent participants who negotiate and cooperate according to their resources, characteristics and common goal.

	Security requirements	Fulfilling Technologies
8	Message level audit trail	Logging, audit trail policy
7	Message level accessibility	SOAP Firewall
6	Message level access control	SAML, XACML
5	Single-sign-on	SAML, Liberty Alliance, .NET Passport, WS-Federation
4	Message level non-repudiation	XML-DSIG, WS-Security, XKMS
3	Message level user authentication	XML-DSIG, WS-Security, XKMS, SAML
2	Message level data integrity	XML-DSIG, WS-Security, XKMS
1	Message level data confidentiality	XML-Encryption, WS-Security, XKMS

Table 2.3: Web Services Security (SOAP message security) Stack

The pre-condition of a successful business federation is the achievement of global security objective and a 'full lifecycle' protection of corporation patrimonial value. Without a central authority to configure and manage security goal, each participant must define its concerns, using policy to express its security profile and requirements. To fit the life-long protection requirement the policy should express issues about "allowed way of usage upon corporate asset" to support end-to-end scale corporate patrimony protection. Several issues have impacted such a 'policy-based' end-to-end security management methodology.

2.3 Toward End-to-end security for collaborative system

End-to-end security for collaborative systems can be characterized by two questions: 'Which partner can access my assets?' and 'What can he do with the assets?' Such a system can be seen as an access control system enhanced with the capability to manage 'due usage' control - ensuring appropriate use of resources, in the way expected/allowed by the owner [154]. In this system, partners' security profiles are summarized as security attributes, which serve as criteria for access decision. By this way, a coordination of partners'

security configuration is established. Furthermore, the impact of complex business process should be considered. The effect of access decision in this system is not just 'permit access' or 'deny access'. It can be any type of consumption activity (e.g. access service, copy data, render multimedia file, etc.) or obligations. The enforcement of this access decision can not be done solely on the resource provider side. It relies heavily on the inspection of the consumer side system.

To support end to end security management that is adapted to any collaborative scenario can be too ambitious. For example the systems for Cloud Computing and Supply Chain Information Management system can be different at either the policy expression level or the enforcement level. Nevertheless, some common traits can be summarized and we can build a system that serves as bedrock for coping with the different scenarios. To achieve this goal, we firstly look at relevant issues. Basic requirements have related technics or ideas in industry or academic field that can be adopted or can be inspiration.

2.3.1 DRM: control of resource usage

First of all, to deal with 'due usage control', one can consider the Digital Right Management (DRM) service. DRM integrates rights and usage control beyond ownership boundary. It has been successfully introduced to protect music or video digital rights by providing a license that allows a reduced use of the content.

2.3.1.1 Definitions

Digital Rights Management is "the description, identification, trading, protection, monitoring and tracking of all forms of rights usages over both tangible and intangible assets including management of rights holders relationships" [136]. It refers to controlling and managing rights to digital intellectual property [256]. It involves the description, layering, analysis, valuation, trading and monitoring of the rights over personal or organization's assets, both in physical and digital form, including tangible and intangible value [255].

The DRM functions can be split into two groups, as depicted in figure 2.8.

Figure 2.8: The two parts of DRM

DRM is used for managing. Providers need to identify their content, to collect metadata related to the content, so that potential customers can find what they want to get. Providers assert which rights upon the content they will release through Rights Expression Language (REL). They also need business models for distributing their assets. Another distinguishing feature about DRM is that it comes up with mechanisms to enforce the released rights, confining consumer's activity to the scope defined by the provider.

2.3.1.2 Architecture

Two different visions from DRM can be presented: an architectural view and a functional view. From an architectural view, three major components can be identified: the content server, the license server, and the client [88]. The content server stores and manages digital contents, information about products (services) that the content provider wants to distribute, and the functionality to prepare a content for a DRM-based distribution. The license server is responsible for managing licensing information. Licenses contain information about the identity of the user or information on the device that wants to use rights concerning the content, identification of the content to which the rights apply, and specifications of those rights. The client resides on the user's side and supplies the following functionalities: DRM controller, rendering application and user's authentication mechanism [256].

From a functional point of view (figure 2.9), the functionalities of the components are as follows:

- Content Provision is the interface used by the content providers to register their digital objects.
- Content Safekeeping stores the digital object in plain format or in a security wrapper.
- License Phrasing (or Offer Creation)is a process that can be seen as

rights metadata provision. The result of this process is a license written in a Rights Expression language.

- Booking refers to the process (and supporting components) of requesting the content. It usually involves a payment process.
- Content Preparation comprises usually the functionalities of watermarking, compression, encryption, enrichment (with metadata) and wrapping.
- Content Distribution can be done by a register center (E-commerce shop, for example), peer2peer networks or unstructured ways (superdistribution).
- Authorization is sending the content key (license) to the consumer.
- Content consumption involves a 'rights enforcement point' that checks whether the requests for accessing the content can be granted according to the license. It also involves a 'content player' that renders the content in a compliant way within the license.

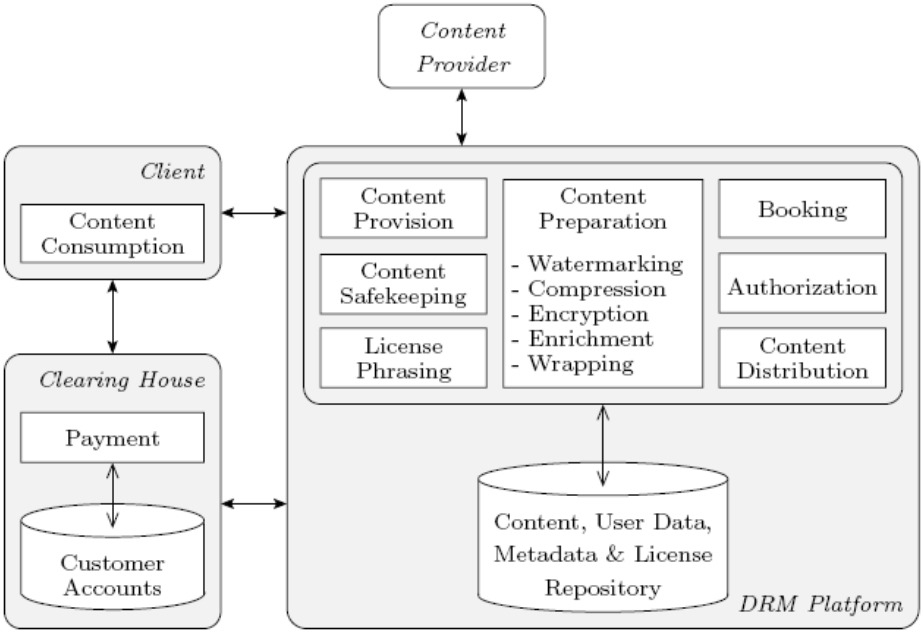


Figure 2.9: A sample DRM system

Several key mechanisms are critical for the DRM, we give further introduction of them in the followings. They are also necessary or inspiring for

building a general DRM system for corporate information system.

2.3.1.3 Authentication

As a foundation for trust, a DRM system must enable authentication for identifying the user and the message exchange. Authentication is usually provided by a certification center. With public key cryptography, the authentication of communication participants is possible without exchanging any additional information. This involves a Certificate Authority (CA) server which is responsible for the identification of participants. By issuing a certificate to a participant, the CA server certified its identity. Furthermore a CA server can issue certificate to other CA servers. Message authentication technologies include symmetric key encryption, asymmetric key encryption and digital signatures to secure communication. Standard protocols as X.509 protocol can be used.

2.3.1.4 Rights Expression Languages (REL)

A 'Rights Expression Language (REL)' provides a mean for expressing rights to digital content (figure 2.10). It's the basic for authorization usages upon asset value.

It should be rich enough to facilitate business models by expressing terms and conditions for digital publications of audio and video files, images, games, software and other digital assets. The application of a standardized REL facilitates interoperability and consistency for DRM systems. Such a language includes different terms:

- Party represents a participant of the business module supporting DRM.
- Asset represents the digital content or service to which the rights apply.
- Rights are described as expressions, granting a given usage or access permissions to digital goods or services. Permissions can be specified with constraints and obligations.

Rights expression language includes a rights vocabulary, or 'Rights Data Dictionary' (RDD), which defines the allowed vocabulary and its semantics in REL instances

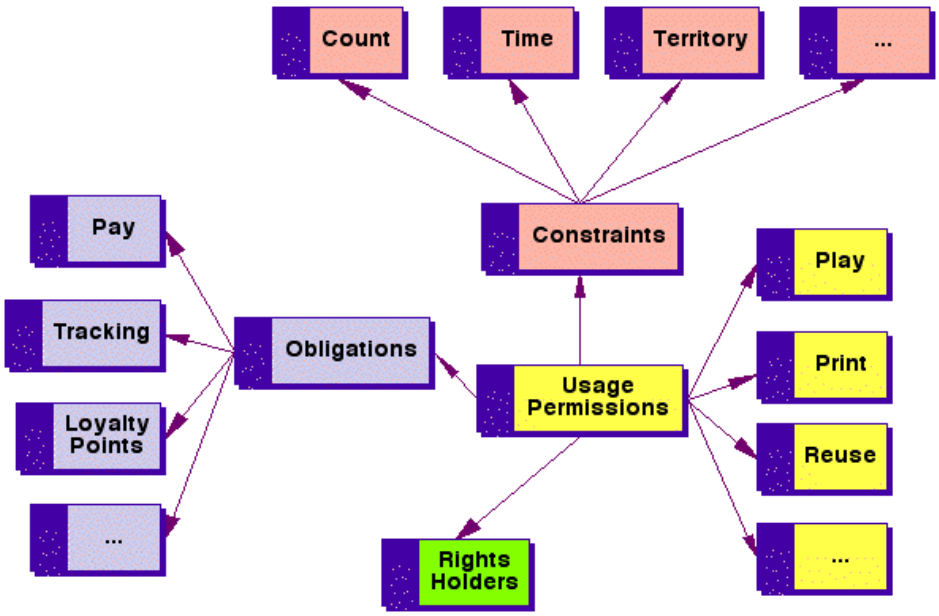


Figure 2.10: A sample structure of REL

2.3.1.5 Content preparation

Content preparation usually includes the content watermarking and the content encryption.

Watermarking is a technique for hiding a piece of information (usually copyright mark) in a data (multi-media, database, stream, etc.), in a resilient manner. It is usually referred to as 'encoding' [276]. This information is used to identify the real owner of a suspect data set, or the source of an unauthorized diffusion [118] (especially, it is named as 'fingerprint' technology), usually referred to as 'decoding'.

Watermark encoding is composed of two main parts. In the first stage, the input data set is securely partitioned into (secret) subsets of items. The second stage then encodes one bit of the watermark into each subset. If more subsets (than watermark bits) are available, error correction is deployed to result in an increasingly resilient encoding [275].

At detection (decoding) time the secret subsets are rebuilt and the individual bits are recovered according to the single-bit mark encoding convention. This yields the original e-bit string. If e is larger than the size of the watermark, error correction was deployed to increase the encoding resilience.

The watermark string can then be recovered by applying error correction decoding to this string, e.g., majority voting for each watermark bit [275].

Watermarking is widely studied and adopted in the area of multimedia [55] [107] and several efforts have lead to solutions for general form of digital work such as relational database [5] [68] [172] [174] [25] [269] [276], stream data [173] and XML [118].

Watermarking certainly brings alteration and distortion of data. This is obviously a limitation for a legitimate purchaser, but it is well known that this distortion is necessary to achieve watermark robustness [174]. Therefore the distortion is limited to some criteria to preserve the quality of data set. For multimedia, e.g. image, the quality can be expressed by signal processing characteristics like signal-to-noise ratio. For other structural data (e.g. relational database, XML document and stream), this is formulated as the preservation of correct query result.

Moreover, general data watermarking has some specificities as opposed to multimedia watermarking. First of all, existing techniques for multimedia can't be applied because distortion metrics, tolerable bounds, and resilience often bear multiple semantics. Secondly, relational database and XML data have more limited 'bandwidth' for watermark insertion. Thirdly, synchronization required for watermark detection is eased by the strong structure of existing keys within databases and XML documents. Lastly, internal correlation should not be assumed for relational databases since tuples can be arbitrarily reordered.

In their pioneering works, authors of [5] [6] defined several properties for watermark systems:

- **Imperceptibility**: The modifications caused by marks should not reduce the usefulness of the database;
- **Robustness**: Watermarks should be robust against degradation caused by either benign updates or malicious attacks;
- **Accuracy**: An owner should not detect her/his watermark in someone else's non-pirated database ('false hit');
- **Incremental updatability**: As the owner adds/deletes tuples or modifies the values of attributes, the watermark values should only be recomputed for the added or modified tuples;
- **Blind system**: Watermark detection should not require the knowledge of either the original database or the watermark;
- **Public system**: The method used for inserting a watermark is public,

the defense lying only in the choice of the secret key.

Relational database watermarking technologies usually choose the attributes with a numeric type to be the encoding channel, given the setting that trade-off of accuracy is tolerable. Authors of [172] proposed a watermarking method for vectorial geographical databases, where the embedded watermarks survive common geographical filters as well as several deliberate removal attempts. The well-known AHK algorithm [5] embeds the watermark bits in the 'least significant bits' (LSB) of selected attributes of a selected subset of tuples. A secure 'message authenticated code' (MAC) is computed using an owner-chosen secret key and the tuple's primary key. The MAC is used to select candidate tuples, attributes and the LSB position. The work of [269] improves the watermark resilience by using multiple attributes and formulates the process as a constrained optimization problem that maximizes or minimizes a hiding function based on the bit to be embedded. Their data partitioning technique does not depend on special marker tuples and is resilient to synchronization errors. The majority voting technique is also used to improve the watermark decoding.

In [277], a watermarking scheme for sensor streams is proposed, where streams are defined as continuous sequences of numerical values. Watermarking is performed by altering salient points of the stream.

Other works [68], [138], [214], [333] address watermarking XML information in various contexts, where watermark embedding values are located through the use of specific XPATH queries. The work of [138] considered structural modification as bandwidth for watermarking.

Watermarking of XML stream has been explored by [173]. They introduce 'local dependencies' between parts of the data stream which are only detectable by the secret key owner. For doing this, they identify two relevant parts of the stream. The 'unalterable' part can not be altered by any attack without destroying the semantics of the stream. The 'alterable' part is still useful for the application, but can be altered within reasonable limits. A finite portion of the unalterable part, combined with a secret key known only by the data owner, is used to form a 'synchronization key'. A non-invertible (cryptographic) pseudo random number generator, seeded with this synchronization key, determines how the alterable part of the stream is modified to embed the watermark.

The content is encrypted using symmetric or hybrid encryption schemes before it is loaded onto the web or streaming server, containing (a) metadata

describing the content and (b) the conditions and prescriptions for decryption.

2.3.1.6 Right enforcement

In DRM architecture, right enforcement refers to the safeguard of content on the consumer side, the execution of access control defined by REL, the monitoring of obligation conformance and the reputation recording/forensic measurement. The content safeguard and access control can be carried out by a system module residing on the client side, whilst the monitoring and reputation can be managed by a third party service. A secured content container is usually achieved by encryption, with similar mechanisms to content preparation.

An encryption scheme consists in two major parts: 'cryptographic algorithm' and 'key' [23]. A cryptographic algorithm is a mathematic function used for encryption and decryption. A key is a secret information that is used by the cryptographic algorithm for encryption and decryption. There are two general types of key based algorithms: symmetric and asymmetric algorithms. Symmetric algorithms use the same key for encryption and for decryption. Hence, sender and receiver have to agree on a secret key that must not be revealed to outsiders in order to enable a secure communication amongst them. Asymmetric algorithms, also known as public key algorithms, use two different keys. One of these keys is called the "private key" and must be held secret, while the other key is called the "public key" and must be published. The secret key cannot be derived from the public key.

2.3.1.7 Gaps to a general usage control for corporate IS

DRM system's components are coordinated to provide a life-long protection to digital content in a DRM perspective:

- 'Watermarking' and 'encryption' are used for content preparation.
- REL is used to control content accessing.
- 'Encryption' and 'decryption' secure the content.
- The 'watermarking' technology is also used for tracking the content re-distribution.

The DRM solution is developed for multimedia industry, but it has the potential to fit the scenario of corporate information system and asset man-

agement, provided that some adaptations and extensions are made. For example, the content securing technologies have been adopted in areas outside multimedia industry, the REL inspires a fine-grained security policy model accommodating the 'usage control' issue, etc.

However, some gaps can be evident between the functions provided by a multimedia DRM system and the requirements of a general (collaborative) corporate information scenario. Firstly, in multimedia DRM, the 'usage control' phase relies on a 'player' trusted by the provider, allowed to 'exercise' the 'usage rights' in behalf of the consumer. However, the usage of digital asset by the consumer system in a general business case can not easily be monitored using the same approach as the one developed in multimedia industry.

Another gap is due to differences between the REL and a general scenario based 'usage control' policy model. The elements to be expressed by a general 'usage control' policy are closely related to the application domain. Due to the variety of the application domains, the number of the elements is vast. Even if we consider only the basic elements, it will concern the whole software stack of IS (see section 2.2.2). This is out of the scope of a standard REL.

Furthermore, DRM systems cover only the scenario of 'one-to-one' interaction by now. Recently we evidence a trend of business federation, e.g. Supply Chain Management, SOA, GRID, Cloud Computing, etc. Such collaborative contexts require re-thinking the mechanisms for an end-to-end protection of corporate asset value.

In the following sections, we discuss the methodologies and tools that may be used to fulfill this gap, identifying what have been done and what is missing for a solution of end-to-end security in collaborative context.

2.3.2 Usage control policy

Several access control models have been proposed recently on building a 'usage control' system for the general corporate IS context [326] [238] [325] [224]. 'Usage control' system has two salient features. First, the system entities are described with their attributes. Usually we can find concepts of 'subject attributes', 'object attributes' and 'system attributes'. Second, the access decision allows variable usage actions upon the object to be granted (or denied), forming an enriched 'Rights' part, compared to the traditional access control scheme which only expresses the grant/deny of 'access' action.

The representative work in industry is the XACML [224] standard. It

is a general purpose attribute-based access control language using the XML syntax. Access decision is based on the status (described by attributes) of the object, subject, context and actions. It is also possible to include the information in the request context that are extracted by XQuery [299] as part of the 'condition'. This XML based syntax makes it easy to be coordinated with other XML technologies, or to be processed with XML based tools. For example, it includes functions standardized in XQuery and XPath [300] to build the attribute predicates in the rule. It is processable with Java XML packages as DOM, JDOM and SAC. An open source negotiation engine is also provided [287].

In the academic field, many works have contributed to the definition and formalization of 'usage control' scheme. In the $UCON_{ABC}$ model [326] [238] [325], general attributes include not only persistent attributes such as role and group memberships, but also mutable usage attributes of subjects and objects. Especially, the authors define models for 'attribute-update' actions, which capture the semantic of 'usage' right. With the grant and exertion of a 'usage' right, multiple consumption actions (e.g. playing several songs) can happen. During this process, the attributes of the object (the amount of the 'not used' objects) and subject (e.g. balance in her/his account) are constantly changing. The continuous enforcement of rights (e.g. deciding when to 'revoke' the right) requires checking such attributes. The works of [325] [156] extend the core $UCON_{ABC}$ model with continuous usage sessions to increase expressiveness of obligations in $UCON_{ABC}$.

Manuel Hilty et al. propose a usage control policy language [130] based on the analysis of usage control requirements and existing control mechanisms [131] [247]. In their work, the usage action is labeled as 'Black-box' and 'White-box' actions. Black-box usage is characterized by data that are subject to usage control which can't be interpreted in any way (that is, it treats the data as meaningless sequences of bits). Black-box usage includes the management and distribution of data. White-box usage, in contrast, interprets data: rendering it, processing it (which includes data modification), or, in the case of programs, executing it [248]. Conditions are specified as time, cardinality (e.g. how many times an action can be performed), events that happen, purpose and static environment.

These works form the bedrock for the 'usage control' policy design. They have great influences in industry and academic fields. Nevertheless, some limits still exist, which restrict their usability for end users. First, in order to fit the application domains, some issues closely related to commercial

scenarios should be incorporated. For example, access decisions may be based on the 'purpose' of usage declared by consumers. Commercial models like 'separation of duty' or 'delegation' should be supported. Second, a knowledge base comprising the vocabulary to describe domain knowledge is necessary. Lastly, users should be certain of the effect co-achieved by multiple rules or policies. In the following subsections, we introduce the existing researches related to these aspects.

2.3.3 Attribute based access control

'Usage control' policies are based on the 'Attribute based access control' model, which is at a dawn stage and under development [303] [9] [304] [324] [60]. A representative work is the XACML language. In such policies, conditions for granting access are based on a set of attributes (compared to traditional models, which are based on one attribute of the entity, such as identity or role). This attributes set identify a set of entities for which the policy applies (see section '4. Examples' in XACML [224] specification). Such policy model has flexibility for defining policies for entities in an open context.

A rule in an access policy language can be seen as a logical expression. In attribute-based access control, a rule is based on the combination of attribute predicates with logical operators (namely, 'AND', 'OR', 'NOT', etc). An attribute predicate usually defines a value scope for a given attribute. For example, '$render\_times > 5$' is an attribute predicate, where '$render\_times$' is the attribute name (of an attribute describing the 'render' action) '5' is the attribute value, '>' is the operator define the value scope of the attribute. Such operators are called 'predicate' in XACML and SWRL, in the sense that they are used to build a predicate for logical expression.

The works of Agrawal et. al [4] and Lin et. al [188] use the following categorization of the predicate operators (in their work, an attribute predicate is called a Boolean expression in the sense it results a Boolean value 'True' or 'False'):

- Category 1: One variable equality constraints.
  $x = c$, where x is a variable and c is a constant.
- Category 2: One variable inequality constraints.
  $x \rhd c$, where x is a variable, c is a constant and $\rhd \in \{<, \leq, >, \geq\}$.

- Category 3: Real valued linear constraints.
  $\sum_{i=1}^{n} a_i x_i \rhd c$, where x is a variable, a_i, c_i are constants and $\rhd \in \{<, \leq, >, \geq\}$.
- Category 4: Regular expression constraints.
  $s \in L(r)$ or $s \notin L(r)$, where s is a string variable and $L(r)$ is the language generated by regulation expression r.
- Category 5: Compound Boolean expression constraints.
  It includes constraints obtained by combining Boolean expressions belonging to the above categories. The combination operators are $\wedge$, $\vee$ and $\neg$. $\neg$ can express the inequality constraint $\neg(x = c)$.

The predicates are used to define the relation between 'attributes name' and 'attribute value'. During negotiation between request and policy, an attribute predicate in the request is evaluated upon the attribute predict in the policy, if they have the same name. If the attribute predicate in the request deduce the attribute predict defined in the policy, it results into a 'True' judgement, otherwise a 'False' judgement.

It is also possible to apply other operators to the attribute value, in order to change definition of value scope or build a composite condition. As evidence, the attribute predicate $delete\_time < render\_time + 10days$ defines a composite condition where the '$delete\_time$' attribute is defined based on the '$render\_time$' attribute. Here the '+' operator is used to change the time value given by the $render\_time$, to define a value scope for $delete\_time$. In XACML and SWRL such operators are called 'functions'.

A comprehensive collection of these operators can be built based on survey of some recent XML-based languages, i.e. 'XACML' [224], 'EPAL' [15], 'MathML' [297], 'SWRL' [133], 'XPath' [298], 'XQuery' [299] and 'XF' [300], as these languages possess enriched functions and predicates collected from mathematic and logic fields. Furthermore, they are closely relevant to SOA and Web Service, which are widely adopted.

An attribute based access control system should support different operators and various data types (number, string, vector, series, temporal factor, set and bag. etc.), in order to fit to a wide range of application domains.

To summarize, a general library that comprises the functions in these systems can be characterized as follow:

- Mathematical operators: arithmetic, algebra, calculus, vector calculus, sequences and series, elementary classical functions, statistics, linear algebra, sets theory;

- Logical operators: NOT, AND, OR, NOR;
- Set (and its extentions: 'bag' and 'list') operators: 'bag' is one that has repeat elements;
- Datetime operators: extraction (get time), arithmetic (add, subtract, etc.), timezone;
- Conversion operators: data type conversion, data format conversion.

2.3.4 P3P: policy for pledge

Policies are usually used by resource providers to specify who can access the resource, e.g. XACML, RBAC policies, etc. On the contrary, the Platform for Privacy Preferences Project (P3P) [69] is a protocol allowing resource consumer (web sites) to declare their intended usage of resources (information they collect about browsing users). The main content of a privacy policy in P3P is defined as follows:

- which information will the server request from users;
- how this information is used (for regular navigation, tracking, personalization, telemarketing, etc.);
- who will receive this information (only the company which has collected it, third party, etc.);
- how long information is stored;
- whether and how the user can access the stored information.

P3P allows browsers to understand privacy policies of a web site in a simplified and organized way. By setting user's privacy settings at a chosen level, P3P will automatically block any cookies that the users do not want. Additionally, the P3P Toolbox [104] developed by the Internet Education Foundation is beneficial for internet browsers against misuse of personal data such as junk mail, identity theft and discrimination. In short, P3P is a step forward in technology for automatic communication of data with individual privacy management [104]. Using a mechanism similar to P3P, a usage control scheme can allow the consumers to express their 'pledges' about the usage purpose.

2.3.5 Separation-of-Duty

Separation of Duty (SoD) is a high-level security policy which defines that a task must be performed by different users [184]. It has existed for a

long time, for example, in the banking industry or in the military context (sometimes under the name "the two-man rule"). It has been recognized as a fundamental principle in computer security [61].

Two types of SoD policies have been extensively studied. One type is the Static SoD (SSoD) policies, where the SSoD policy states that the permissions to complete a sensitive task should be assigned separately to k users. For example in role-based access control (RBAC), SSOD is enforced by using Static Mutually Exclusive Roles (SMER) [183], which is to declare that several roles are mutually exclusive. In this case, no user is allowed to be a member of all these roles.

The other type SoD policy maintains a history on who performed which tasks. It defines that if a user has performed a given step of a task, it is not allowed to perform other steps. In this case, enforcement of SoD is done before each step is performed by a user. Therefore, this type of SoD is referred to as Dynamic SoD [262] [213] [101] (also called 'dynamic segregation of duties' [101] AND object SoD [213]). Whereas in RBAC, it's referred as constraints of Dynamic Mutually Exclusive Role (DMER): simultaneously invoking several roles in one session by one user is exclusive prohibited (the roles themselves are not SMER) [183]. The SMER and DMER are important constraints that are included in the ANSI/NIST standard for RBAC [13] [95].

In a usage control scenario for collaborative context, Separation of Duty can be achieved in the sense of generalizing the two approaches above. SSOD definition is based on all possible subject attributes, including (but not confined within) 'role' attribute. An sample expression is:

$$\neg has\_attribute(R1, S) \leftarrow has\_attribute(R2, S)$$

where two attributes 'R1' and 'R2' contradict each other.

DSOD is supported through 'event' history of the business process, either in current session, other concurrent sessions or past terminated sessions. Normally (DSOD) enforcement is based on events that happened during current transaction (e.g. one action contradict another) and current state information. It is also possible that one encounters the requirement of SoD based on the tasks a user has performed in past (terminated) sessions or in other concurrent sessions. For example, in order to protect privacy, an information owner (e.g. in a social network) may constrain others from gathering two blocks of information of her/him [232]. In a process of building collaborative enterprise (e.g. in supply chain), a party may specify that parties cur-

rently involved in another collaborative enterprise should be excluded from the collaborative enterprise, in order to mitigate the risk of leaking sensitive information to competitors.

DSoD based on events in past/concurrent transactions is in the form of specifying that any subject S that has exercised action A1 in concurrent/past transactions can not assume right R (A2) upon object O:

$$\neg A1(O) \leftarrow E(A2(O), S)$$

There are few discussions of SoD based on concurrent sessions in academic research. However we believe it can be required in industry due to the development of business federation, e.g. for protecting trade secret in supply chain. Another important commercial scenario is delegation (and revocation), which commonly exists in many industrial areas. A security model should take into consideration this issue.

2.3.6 Delegation and revocation

Delegation is the process whereby a user without administrative prerogatives obtains the ability to grant some authorizations [27], from a user that has the administrative prerogatives.

In the DRM scenario or general commercial scenarios, we can identify two types of delegations: delegation of rights and delegation of control:

- Using 'delegation of rights', the owner of an asset ('object') can define that a subject who holds a 'right' upon the asset can further delegate this right to other subjects. In usage control policies, 'delegation' can be modeled as a kind of (special) 'right'. The object of 'delegation' can be other 'usage' rights. The eligibility of the subject for the 'delegation' rights is decided according to its identity, role or any other attributes.
- 'Delegation of control' is that the owner of an asset delegates the ownership upon this asset to a subject. In this case, the asset is transferred and possessed by the new owner.

Two major parameters of delegation is the 'delegation depth' and 'revocation'. 'Delegation depth' defines how many subjects the delegation chain can have, in other words: how many steps a right can be passed down.

Revocation is the 'reverse-operation' of delegation. It also serves as a parameter of the 'delegation', in usage control policy, to denotes when and how a delegation of rights is 'taken back' by the asset provider.

Revocation can be described by several attributes [301] [26]:

- **Modality: pre-set revocation and post revocation**
  - 'Pre-set revocation': Revocation condition is pre-set by asset provider. It can be defined according to 'time' (at what time a delegation will be revoked) or 'event' (if an 'event' occurs, then the delegation is revoked).
  - 'Post revocation': Revocation is prompted by asset provider during the working process.

  'Post revocation' is more dynamic and flexible, whilst 'pre-set revocation' is more helpful for consumers to avoid the risk of failed federation caused by unforeseen revocations.
- **Dependency: dependent and independent**
  - 'Dependent revocation': only the direct ancestor in the delegation chain can revoke the delegation.
  - 'Independent revocation': any direct or indirect ancestors in the delegation chain can revoke the delegation.
- **Propagation: local vs global**
  - 'Local revocation': revocation applies only to the direct child in the delegation chain.
  - 'Global revocation': revocation applies to all the children in the delegation chain.
- **Resilience: delete and deny** Provided A and B both delegated a Right (Rt) to C:
  - 'Delete-based evocation' means: If A'deletes' Rt from C but B does not delete it then C still has Rt.
  - 'Deny-based revocation' means: If A'denies' Rt from C then C doesn't have Rt, whether B'deletes'/'denies' Rt from C or not.
- **Dominance (Role Resilience)**
  - 'Weak revocation': 'Delete' the rights of a Role only.
  - 'Strong revocation': 'Delete' the rights of a Role and all its senior roles (the roles that comprise rights in current Role).

The detailed analysis of the 16 states caused by the combination of the parameters of revocation is given by [301]. For a usage control scenario,

we suggest the 'PRESET-INDEPENDENT-GLOBAL-DENY-STRONG' revocation mode. This is a 'safe' mode, where the consumer can know the condition of the revocation and assess the risk related to the revocation of a right delegated to it (with 'PRESET'). This mode is also the most restrictive mode, where a delegation is 'easy' to revoke ('INDEPENDENT') and once revoked, it is revoked totally ('GLOBAL', 'DENY' and 'STRONG'). Nevertheless, this mode can not be adapted to all scenarios, due to its restrictiveness. Other modes can also be used when necessary.

A security policy scheme accommodating above factors has rich features and great dynamicity. In order to mitigate the risk of un-expected effect and to make system behavior more predictable, policies and system behaviors are usually described using formal models. We give a brief introduction of these works in the following subsections.

2.3.7 Foundations for policy models

Formal models, as logic systems, are built up with axioms and theorems and possess intrinsic consistency. Using logic models for formulating the semantics of policy models is a natural choice, as the rule effect must be highly predictable. Many works have been done in this policy formulation area.

2.3.7.1 Access Matrix

An Access Control Matrix (or Access Matrix) is an abstract security model describing the protection states in a computer system, by characterizing the rights of each subject on every object in the system [176]. Static permissions in Capability-based security [179] (A 'capability' also known in some systems as a 'key', is a communicable, unforgeable token of authority) and Access Control Lists [106] (which specifies which users, or system processes, are granted access to which objects, as well as which operations are allowed) can be modeled using Access Control Matrices, leading to a two-dimensional array (subjects being the rows and objects the columns). A recent example is the description of ABAC [324]. Access Matrix has been criticized as then can not model dynamic system behaviors [205].

2.3.7.2 Logic foundation

A policy model based on flexible rules system, e.g. RBAC (or RoBAC or 'ABAC') is usually required to handle the dynamic system. The common logic foundations for such policies are First Order Proposition Logic and First Order Predicate Logic. A basic rule in a policy model is in a Horn Clause form, which is a disjunction of literals with at most one positive literal, e.g.:

$\neg p \vee \neg q \vee ... \vee \neg t \vee u$

or written equivalently in the form of an implication:

$(p \wedge q \wedge ... \wedge t) \rightarrow u$.

Here p,q,t,u are predicates (e.g. attribute predicates in 'ABAC') or propositions. Horn Clauses can be either first order or propositional forms.

With computing systems getting more dynamic, powerful and complex, the needs of formalizing system behavior grow (for example within database theory research, logic programming and policy research). In response of these needs, many specific logic systems have been developed and their numbers are growing. In the following we present some works that are closely related to the issues we encounter while defining a collaborative usage control scheme for end-to-end security management.

2.3.7.3 Logics for identity based authentication

There are several approaches for formalizing identity authentication, as summarized in by Peter C. Chapin et al. [53], for example the BAN authentication logic [46], ABLP distributed authorization logic [2] and logic programming. These works have been done mostly in order to address the identity based access schemes (e.g. MAD, DAC, etc), but they are still sufficiently meaningful for more rich-featured systems as RBAC, OrBAC, 'ABAC', UCON, etc. Although accesses in such systems are granted not directly (and only) based on identity (but based on 'role', 'organization', or any other attributes), identification is the underlying mechanism to ensure the rights are granted to the expected subjects.

2.3.7.4 Logics for access control

The basic for the semantic of access control system is logic programming (as datalog [3] [24], prolog [192] [161]). The work of [166] developed a proof-theoretic formalization of XACML using natural deduction rules. This work

formulates the mechanism for matching attribute predicates between a request and a rule. It also describes the impact of rule/policy combination algorithms. The effect of policy, regarding a request, is decided by both the matching of attribute predicates and the impact of rule combination algorithm. Authors of [303] present a framework that models attribute-based access control using logic programming with set constraints.

A major extension of the Access Control policy model to fit the Usage Control policy model is the 'Obligation', i.e. actions that must be performed by subject, attached to a right. Deontic logic [49] [19] [139] provides the foundation for obligation in policy regulation. [85] presents a model where obligations are tied to authorizations. These works will be useful for us to formalize the 'obligation' part of our policy model.

Other security/rights management model as delegation, has been addressed with formal descriptions too. For example, Delegation logic [52] [181] [182] was proposed to formalize the 'delegation' in RBAC. These works can help formalizing effects of a usage control policy. The effects lead to state changes in the system, usually following some temporal patterns. There are many works concerning this aspect of policy model formulation.

2.3.7.5 Logics for formulating system behavior

A formalization dedicated to the $UCON_{ABC}$ model has been introduced [327] [328] using Lamport's Temporal Logic of Action (TLA). It describes the temporal constraint between the actions in Authorization (granting of right), Obligation and attribute update that result from exercising the granted rights or imposed obligations. There are some similar works [142] [249] using temporal logic for formalizing UCON system.

There are also works that analyze system behavior using computation tree logic [63], model checking [21] finite state automata [21] or Petri Nets [157].

These works do not take into consideration the impact of random events in the system. Whereas the work of [72] incorporate Event Calculus (EC) [167], with Abductive Constraint Logic Programming (ACLP), to give the policy the capability to describe events that are regulated by policy (e.g. usage action, attribute changes) and events that are 'spontaneously' generated by the Information System (e.g. session events as start and termination, environment factors). Abductive Logic Programming (as used in [166]) captures the basic rule system semantics. Event Calculus is powerful in modeling

decentralized system behavior, as in service networks [194], pervasive computing [198] or environment monitoring [43]. The **Ponder** [76] [195] [293] policy system is built using EC.

In brief, ACLP explicitly describes the reasoning process when evaluating a request upon a policy. Temporal logic regulates the state change incurred by the enforcement of rights and obligations. EC helps modeling the impact introduced by events from outside the policy model and makes it aware of the context.

One major benefit of formulation is it facilitates verifying whether of effects of policies is as the expectation of policy author, which is an arduous task related to many factors. There are many works that dedicate to this task.

In fact, it is quite usual to incorporate two or more formal model to define a complex system (see chapter 4 for more information) or analysis its characteristics (see the following section). In chapter 4, the semantics of our policy model is described with ACLP (similar to [166]) and event calculus (similar [72]). The behavior of our system complies with the patterns introduced in introduced [327] [328].

2.3.8 Combining rules

A policy system usually consists in multiple rules. It is possible that two rules are defined on a same (or intersected) set of entities (subjects, objects, rights). Then the effects of the two rules should be combined to produce a final effect upon the intersected entities set. Methods for combining rules can be differentiated along two dimensions: 'Specificity precedence' and 'Effect precedence'

Specificity precedence [254] is that a rule applying to a more specific entity set takes precedence. For example, in Operating Systems (Windows, Linux, etc), if a policy P_1 defines that a group of files is allowed to be read by a user U but, at the same time, one file in the group is defined by another policy P_2 as not readable by U, P_2 takes precedence.

Effect precedence [186] is that the precedence between rules is defined by their effect (deny or permit). For example, XACML defines 4 'combinators' according to rule effect, namely 'deny override' (deny rule takes precedence), 'permit override', 'first applicable' and 'only one applicable'. It is also possible to use other types of combinators [186] [217]. Generally speaking, there are 5 ways to combine multiple policies:

- One effect takes precedence, for example 'deny override' and 'permit override' in XACML [224].
- Majority takes precedence, that is the effect (deny or permit) that has more vote (more rules) wins. For example the 'weak-majority', 'strong-majority' and 'super-majority-permit' presented in [186] belong to this category.
- Precedence by order, that is the one which takes precedence is defined by some sorting algorithm, for example the 'first applicable' in XACML [224].
- Consensus: while the former three types of combinators deal with conflicts between rules, this strategy defines that there should be no conflicts between rules in order to get a decision [252] [186]. The work of [186] proposed 'weak-consensus' and 'strong-consensus'. The integration algebra proposed in [252] is based on 'strong-consensus' strategy.
- Matrix: it decides the co-effect of several 'rights' use a matrix [164]. This approach is only efficient when the rights number is fixed and not very large. Otherwise, the matrix gets too complex.

The rule combination problem exists not only in security policy management, but also in trust management for distributed system. For example the work of [312] presents a method for combining trust values and calculating the final trust relation. Rule combining problem is so common that some works discuss 'aggregation algebra' at an abstract level [44] [217] [252]. These works are usually based on a 4-valued logic system (e.g. [2] and [217]) due to the fact that the effects of rule can be 'permit', 'deny', 'indeterminate' and 'not applicable'. Indeterminate denotes the case that the negotiation between policy and request can not be terminated, mostly due to a lack of information or the failure in fetching subject credentials or attributes. Not-applicable is that the request matches no policies. This is usually due to the insufficient 'coverage' of policy set (see the following section 2.3.9 for more discussion).

In practice the 4-valued decision set of rule / policy negotiation result should be 'flattened' into the effect of 'deny' or 'permit'. One way is by simply giving a negative decision (deny effect) for the case of not-applicable or indeterminate (so-called '**negation as failure**'). This strategy is based on the '**negative closed world**' assumption [30] where the 'default' effect for a request is 'deny' if it's not explicitly permitted by the policy set. The other way is the '**positive closed world**' assumption [30] where the 'default'

effect for a request is 'permit' if it is not explicitly denied by the policy set.

2.3.9 Policy ratification

The 'gulf of execution' [221] problem – gaps between human intentions and technical implementation – commonly exists in IS. In security policy management, the so-called '**policy ratification**' [4] (or policy safety analysis [188]) aims at finding out whether a policy system meets the intention of policy authors. It mitigates the policy 'leakage', where rights are granted to a subject that should be unauthorized, or 'over approximation', where a subject expected to obtain the right can never be authorized according to the policy set.

Ratification is conveyed by the analysis of 'coverage', 'conflict', 'dominance' [188] [30] [155] and 'decidability' [74] [32].

2.3.9.1 Coverage

Coverage (or 'totality' [32]) is whether the policy covers the interest case [155] (all possible access requests in the system) considered by the policy author. This involves that each request which may occur is associated to an explicit policy (or policy set) regulating it [4]. To check the coverage capability of a policy system, one can use method of 'policy effect query' in the EXAM system [188]. The coverage problem depends not only on the policy model and the experience of the policy author. It also depends on the capability of policy vocabulary for describing the knowledge of the application domain.

2.3.9.2 Conflicts detection

Two policies are in conflict, if their effect cannot be achieved simultaneously [4]. If this occurs, we say that the 'consistency' is not maintained [32]. Conflict is possible only when there is **'policy overlapping'**, that is, two (or more) policies are defined on a same (or overlapped) set of entities (e.g. 'subject', 'object', 'context' and 'right' in usage control scheme). An example of conflicting policies may look like: $(\neg Drink(alcohol) \leftarrow Age(X) < 18) \wedge (Drink(alcohol) \leftarrow Age(X) > 16 \wedge Gender(X) = Male)$

The overlapping detection in attribute-based policy model is a bit more complex than in identity-based models (e.g. RBAC, DAC, MAC). Identity-

based models use one token (identity, 'Role-identity' assignment) to designate the subject, whereas the attribute-based model uses multiple tokens (attributes) to mark 'a range of' entities (a good example is OWL, in which a set of 'properties' is used to define an entity). Consequently, deciding two definitions of entities, say X and Y, overlapping involves examining through the two sets of attribute that describes them. If each attribute definition of X has a corresponding attribute definition for Y (i.e. they have the same attribute names and the attribute value range of the former covers that of the later), we say that X 'covers' Y, given X and Y belong to the same category (A 'category' can be 'subject', 'object', 'context', 'right' or 'obligation').

There are many discussions about policy conflicts during the past years [207] [188]. For example, [207] gave a detail categorization of the conflicts on policy modality and goals. Several of them have great impact on our policy aggregation design. Based on these works, we summarize the main conflict types for our work (that will be discussed in chapter 5).

- **'Positive-Negative Conflict of Modalities'**: It occurs when a subject is both authorized and prohibited for a right on an object.
- **'Conflict between Imperial and Authority Policies'**: It occurs when a subject is required to carry out an action by 'obligation' of a policy and prohibited to carry out this action by another policy.
- **'Conflict of Priorities for Resources'**: It happens when two (or many) exclusive usage requests are made to a resource, or there is a limited amount of a resource, which is exceeded by the request summation.
- **'Conflict of Duties'**: It indicates the situation that two actions are not allowed to be performed by the same subject, which leads to the 'Separation of Duties' principle [262].
- **'Conflict of Interests'**: It means the same subject is not allowed to perform a same action upon two objects.

2.3.9.3 Dominance

A policy (or a set of policies) X is dominated by another policy (or a set of policies) Y, if that adding X to the system does not affect the system behavior governed by Y [4]. Then X is called 'ineffective' [4] or 'redundant' policy / set of policies.

Dominance is also based on entity overlapping. For example, in our previous policy example '$\neg Drink(alcohol) \leftarrow Age(X) < 16$', the policy '$\neg Drink(alcohol) \leftarrow Age(X) < 16 \wedge Gender(Y) = Male$' is redundant.

An obligation Ob_A is under the dominance of another obligation Ob_B, if the fulfillment of Ob_B implies Ob_A. In such case Ob_A can be deemed as redundant. As the usual form of obligations is a condition that must be fulfilled (e.g. actions) according to some temporal constraint, Ob_B implies Ob_A means:

- Firstly, conditions in Ob_A are in the scope defined by Ob_B. For example, $within(delete(A.a), oneWeek)$ is implied by $within(delete(A), oneWeek)$, where '$A.a$' is a part of data '$A$'.
- Secondly, if the condition of Ob_A has a temporal constraint, the temporal constraint should be in the scope of the temporal constraint in Ob_A. For example, $within(delete(A.a), twoWeeks)$ is implied by $within(delete(A), oneWeek)$.

Such a method for detecting obligation dominance has been proposed recently [216] and has been discussed by other works as [30] [155].

In summary, policy ratification (or policy safety analysis) is the foundation for Collaborative Usage Control (CUCON) policy model, where multiple policies co-effect the object produced by the business federation. Conflict detection is the most important analysis that affects the usability of the CUCON policy.

When applying to a specific application domain, the usability of CUCON policy is affected by the capability to represent domain knowledge. This concerns the definition of the 'vocabulary base' of the policy, which is used to describe domain knowledge.

2.3.10 Domain knowledge representation

An access control (or usage control or even DRM) scheme in real world application consists in both policy model that defines (according to logic foundations) the policy 'grammar' and semantics, and the 'vocabulary' used to describe domain knowledge. The grammar is defined in a context-free level and describes the reasoning mechanism independently of specific application domain. The vocabulary base bridges grammar with an application domain. These two aspects co-define the expressing capability of a policy scheme.

For example, in the usage control scheme proposed by DOCOMO euro-lab [130] 'usage' actions are classified into 'Black-box' and 'White-box' actions. It also includes, as 'conditions', time factors, events, cardinality (e.g. how many times an action can be performed), purpose of usage, and environmental attributes.

The work of [74] defines a set of 'context' factors in order to describe different usage control scenarios, namely 'temporal', 'spatial', 'user-declared', 'prerequisite' and 'provisional' contexts. The definition of 'temporal' and 'spatial' elements depends on the time of action and the location of subject, separately. The 'user-declared' context depends on the subject's objective (or purpose). The 'prerequisite' context depends on characteristics of the subject, the action, the object or the environment, e.g., a role '$Physican$' or a premise '$Office$'. The 'provisional' context depends on previous actions the subject has performed in the system.

There are some requirements for the vocabulary base concerning the characteristics of collaborative system to set the security level contract:

- It should have an extensible'vocabulary' allowing new domain/industry specific knowledge to be described easily;
- It should be interoperable as the federation context can span multiple organization boundaries;
- It should be machine readable and processable in order to be adapted to large scale complex application;
- It should ensure consistency during the process of federation.

Recent years have seen many works that applies ontology technology to security policy system to capture domain knowledge, e.g. [60] [97] [150]. An ontology describes the concepts in a domain of interest and also the relationships that hold between those concepts [132]. So the knowledge represented with ontology can be used to reason about the entities within that domain and find new knowledge, thus describes the domain.

The Web Ontology Language (OWL) is a widely adopted knowledge representation languages for ontology construction. Basically, OWL presents concepts in a structured way, defining the Class-Subclass relation (i.e. 'property' in protégé [132] [114]). It is designed based on Description Logic [212] [18] [165] , a tractable subsets of First Order Logic (FOL) by formal semantics. It allows the use of a reasoner to check whether all the statements and definitions are mutually consistent. OWL has a rich set of operators - e.g.

intersection, union and negation (see the Protégé toolset [132] [114] for examples). By applying these operators on concepts, one can build up complex concepts with basic ones, using the deductions process driven by the 'reasoner'. Properties of a concept are described thanks to basic data type as numerics, duration, chars, etc (for example the protégé 4.0 toolset provides 47 types of 'data'). Moreover, the OWL allows defining terms to describe arbitrary relations between concepts, besides the basic 'Class-Subclass' relation. It has defined some characteristics on relation properties, namely 'Functional', 'Inverse functional', 'Transitive', 'Symmetric', 'Asymmetrric', 'Relexive', 'Irreflexive'. Deductions and consistency check are based on these characteristics.

Nevertheless, OWL can only address a rather small problem space: reasoners based on OWL can only deduce Class-Subclass relation. This limit can be overcome with SWRL [114]. The Semantic Web Rule Language (SWRL) [133] combines sub languages of OWL (namely, OWL DL and Lite) with those of the Rule Markup Language (Unary/Binary Datalog). Rules are defined as implications between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent holds, the conditions specified in the consequent must also hold [114]. A simple example of these rules would be to assert that the combination of the hasParent and hasBrother properties implies the hasUncle property [133], which could be written as:

$$hasParent(?x1, ?x2) \wedge hasBrother(?x2, ?x3) \rightarrow hasUncle(?x1, ?x3)$$

which in abstract syntax is written as:

$$\begin{aligned} Implies(Antecedent(hasParent(I-variable(x1)I-variable(x2)) \\ hasBrother(I-variable(x2)I-variable(x3))) \\ Consequent(hasUncle(I-variable(x1)I-variable(x3)))) \end{aligned}$$

Using these language tools, on can create SWRL rules that use the vocabulary of an OWL ontology and reason in a semantically consistent way. It takes advantage of both the ontology and the rule base knowledge to draw inferences (see reasoners such as Pellet [62], Racer [124], Fact++ [228], Hermit [119], etc). These inferences add new facts (add relations to concepts) to the knowledge base. Being a rule inference language, SWRL can even be used to express security policies or other user preferences, based on vocabulary defined with OWL. There are some recent works following this approach [60] [97] [150] [246], aiming at, e.g. managing the Web Service QoS [50].

2.3.11 Fitting the federation context

Security management for a federated business process requires capturing the dependency relation between services provided by the partners, in order to coordinate the different partners' security profiles (requirements and attributes). For example the works of [81] [159] [160] model complex Information System with Architecture Analysis and Design Language (AADL), whereas the IT industry may use the WS-BPEL language to model business processes. Methods dedicated to the dependence analysis between services are necessary to manage usage control (w.r.t. data and processes) policy.

2.3.11.1 Service dependency

In business federation, assets transfered across organization boundaries can be merged with other assets. In order to give a full lifecycle protection to an asset, it's necessary to capture the assets derivation relations and track the asset in the business process artifacts. This issue is analogous to the 'Program Slicing' [117] [331] based on System Dependency Graph (SDG) [117] [122]. Program slicing asks about which statements influence the current statement under exam (backward slice), or which statements are influenced by the current statement (forward slice). A SDG sample is illustrated in figure 2.11.

PDGs are interconnected thanks to call statement to form a SDG. The entry of a PDG is represented by an 'entry' vertex and a collection of 'formal-in' and 'formal-out' vertex, representing the parameters inside the procedure for carrying the input and output data.

In the calling procedure, a 'call vertex' represents the call statement. The actual-in and actual-out vertex attached (by control dependency) to the call vertex represent the parameter in the calling procedure for carrying the input and output data. Program slicing is efficiently computed by reachability analysis in the program's SDG.

We will use a similar approach to build the Service Dependence Graph and use queries on it to capture the assets aggregation pattern (see chapter 5). To fit this goal, the way services and assets are composed must be taken into account.

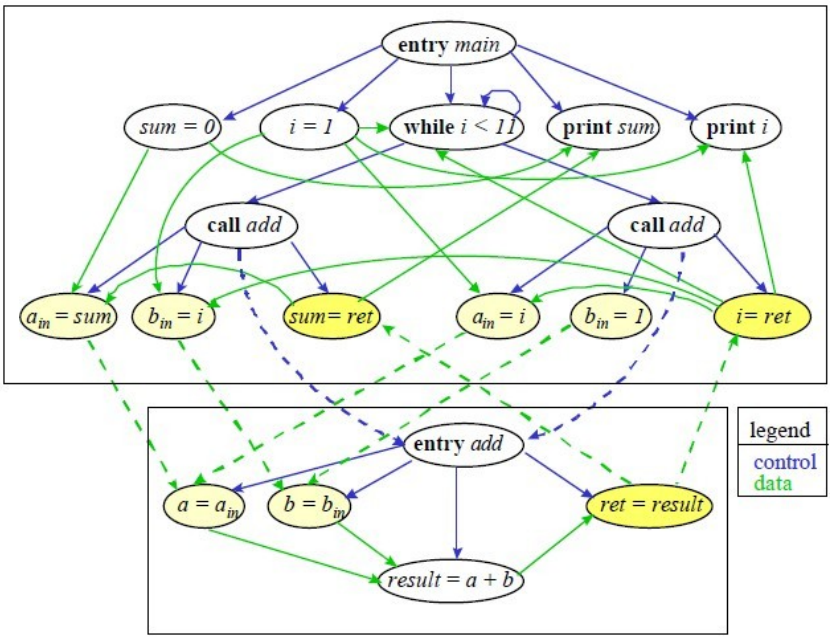


Figure 2.11: System Dependence Graph [117]. *SDG is a collection of Procedure Dependence Graphs (PDGs) [96] [96], each PDG representing one procedure. The vertices of a PDG represents the individual statements (e.g. 'sum=a+100') inside the procedure. The edges presented by green lines stand for 'data dependency', where the value assignment in the vertex at the start point of the edge can be referenced in the vertex at the end point of the edge. In other words, the* data *in the statement at the end point of the edge depends on the data in the statement at the start point of the edge. The blue lines represent 'control dependency' between statements, the* execution *of the statement at the end point of the edge depends on the statement at the starting point of the edge.*

2.3.11.2 Service composition pattern

As provider's security requirements must be maintained during the business federation, the way assets can be converted, split or merged must be captured. In other words, the security level agreement is evolving according to the business process organization. Some works [199] [322] [229] [190] [180] have analyzed the process patterns in SOA, leading to 4 types:

- **sequential pattern**:

pattern 1: invoke

pattern 2: receive

pattern 3: invoke/receive

pattern 4: non-occurrence (the 'not happen' of an event)

- **relayed pattern**

pattern 5: request with referral (A request B to respond to C)

pattern 6: relayed request (C request A to further request B)

- **recursive pattern**

pattern 7: circular invoke (A request itself)

pattern 8: multi-receive (A request B once and receive multi-response)

pattern 9: contingent invoke (if B didn't respond, A request C)

- **parallel pattern**

pattern 10: one-to-many invoke

pattern 11: one-from-many receive, synchronous (A is activated if all its preceding services have been completed)

pattern 12: one-from-many receive, asynchronous (A is activated if some of its preceding services have been completed)

pattern 13: one-to-many invoke/receive

pattern 14: dynamic routing

Among these patterns, the parallel pattern rise most concerns during the context management process (see chapter 5). Depending on these patterns a dependency graph can be built to identify context management impact on the policy 'propagation' among the business process.

2.3.12 Usage enforcement

Usage control enforcement usually relies on a module deployed on the consumer side system used to monitor the sensitive information flow that carries the providers' asset value. For example, according to a provider's policy it may 'lock' any operation that lead to write a file to disk (i.e. data storage) or socket (leading to information leaking through network). This can be done by inspecting and blocking system calls. Several recent works have proposed architectural design for usage enforcement at the OS level [267] [289] [10] [306], application level [326] or Web Service mediation level [7] [128] (e.g. ESB [112] [111]). Most of these works, however, don't give enough technical detail about how consumer's operations upon assets are monitored and checked according to policies.

Authors of [129] give an in-depth discussion about monitoring usage control by inspecting system calls generated by the consumer process. They gave a holistic view of the consumer system states that can incur policy breach. Consequently, all the operation patterns leading to such states should be blocked. They also propose a method for translating high-level usage control policy to low-level operations, so that the monitoring component can compare consumer operations with these low-level operations.

Mapping high-level policies to low-level policies recently draws many research attentions [330] [71] [70] (some called 'policy refinement'), as it is a necessary stage between policy definition and enforcement.

According to literature [129], system call based information flow tracking suffers from the so called 'over-approximation' problem: after a rather long-term run of consumer process, during which it exchanges information with other components of the system (file on disk, memory area, socket, other process, etc), it is possible that all these components are deemed potentially consisting the sensitive information. To overcome this limit, one may require an approach at a more detailed level that 'looks into' the consumer process (which deals with the sensitive information) and inspects the instructions generated by this process.

The usage monitoring is highly related to the research of 'Information Flow' tracking, where the purpose is to track (and block) the flow of sensitive information from one (trusted) component of the system to another (untrusted) component [113] [89] [128] [1] [257]. This can be implemented at the OS level by tracking information flow through system interfaces (e.g. system calls) [113] [89] [128] [1] [257]. It can also be done at programming

language level by analyzing code with System Dependence Graph [331] [121] [122], by directly enhancing the code with labeling [332] of type [105], w.r.t. information flow control perspective, or by attaching meta-policy to the code for information control [272] [175].

In some circumstances, e.g. when a piece of information is switched to disk or saved to a file on disk, it is possible that the user can 'bypass' the usage monitoring mechanism implemented on monitor consumer process, e.g. by directly access (normally or hacking) the file on disk. To cope with this circumstance, a more holistic monitor mechanism is needed. A possible approach is through monitoring system log, using methods similar to the works of [315] [313] [314].

Another important issue is that the monitoring module needs to be sure that the system calls are not modified. Otherwise, operations executed by the modified system calls are not as expected, the monitoring will be ineffective. In order to insure the integrity of the target system, the 'Trusted Computing' technology [10] can be used. 'Trusted Computing' technology establishes trust through the software stack of the target system based on a Trusted Platform Module (TPM) [120], a chip embedded on the target system. The TPM usually incorporates the following functional components [108]:

- asymmetric key generation, encryption and digital signature capabilities;
- SHA-1 hashing engine;
- random number generation;
- several Platform Configuration Registers (PCRs) for recording platform state;
- a means of reporting the state to remote entities ('remote attestation' [260]);
- secure volatile and non-volatile memory.

Together with a 'Core Root of Trust for Measurement' (CRTM, which can be contained within the BIOS Boot Block), TPM can be used to measure the integrity of the target platform. An integrity measure is the cryptographic digest (or hash data) of a piece of code [241]. During the 'trust bootstrap' (of authenticated boot process), the CRTM first generates its integrity measure coupled with the one of the BIOS (POST-BIOS). This measure is sent to TPM to be recorded in the first PCR (PCR-0, of the 16 PCRs). Then the control is passed to the POST BIOS, which generates measures of the

platform configuration like ROM code, OS loader, etc. These measures are sent to TPM and recorded in PCR 1-5, before the control is passed to the OS loader. Then at each stage of the OS loading process, measures are recorded in Stored Measurement Log (SML), maintained externally to the TPM. The generated measures can be compared with a set of known measures to check integrity. By this way a chain of trust starting from hardware level to OS and application level can be built, where the previous executing code check the integrity of the next components (a piece of code) to be executed [270].

Another important functionality of TPM is to provide secured storage. TPM contains a Storage Root Key, a 2048-bite key pair for an asymmetric encryption scheme [108]. It can be used to encrypt data (TPM protected data object) or other keys (TPM protected key object). To ensure integrity of the encrypted data, a 20 bytes 'authorization data' can be associated to it before encryption. The 'authorization data' is checked before decryption. If decrypted data has been tampered with, the authorization data will most likely be corrupted [108]. TPM can also be used to protect integrity of Virtual Machine [250] [56] [286] [109], Agent [271] [210], GRID [334] [321] or p2p environment [264]. Usage control in these systems can uses TPM as a trust root.

2.4 Collaborative usage control system requirements

The purpose of our research work is to develop an access/usage control system that comprises variable security factors and coordinates multiple partners' activities w.r.t. these security factors. Based on the previous discussions we can summarize the criteria for deciding how such a system can fit the requirements of collaborative context:

- **Usage control**: A traditional access control policy usually grants only 'access' (read) rights. A usage control policy has a richer 'rights' part able to grant various 'usage' (consumption) activities.
- **Policy model**: The most influential policy models are Identity (credential)-based models (MAC, DAC, etc.) and Role-based models (RBAC). Recently Attribute-based model (e.g. XACML) emerges. It can incorporate various attributes of the subject (consumer), object (resource), or

context (environment, business context, etc.), so that several security factors can be integrated in a usage / access control system.

- **Policy aggregation**: A usage / access control system for collaborative context should be able to manage policy interactions due to the communications between partners. In some contexts, policy aggregation is also required for managing resources subsidiary to more than one partner.
- **Context management**: In a decentralized / collaborative context, the policy system should be able to manage the resource exchange paths among partners (e.g. tracking information flow), in order to maintain coalition between policies and assets aggregations.
- **Vocabulary base**: A complete policy system should provide a vocabulary base for describing application domain knowledge, for users to compose their policies based on it.
- **Negotiation Engine**: Its basic functionality is to decide whether a request can be granted according to policies or not. In a collaborative context, it should manage policy interaction and policy aggregation.
- **Enforcement Mechanism**: A complete policy system should have mechanisms to enforce the decision taken by the negotiation engine to the partners' systems. Usage control policies requires much more comprehensive enforcement mechanism than traditional access control policies, as usage control involves monitoring the consumer side activities (consumptions) upon resources.

$UCON_{ABC}$ scheme [238] [326] [237] [325] [329] [327] [328] fits a part of these goals as one of its salient features is that the usage process is decomposed, allowing a detail discussion of the models relating to obligation enforcement (i.e. *pre-obligations*, *ongoing-obligations*) and attribute update (i.e. *pre-update*, *ongoing-update* and *post-update*). Temporal constraints between these models and usage actions are also formulated [327]. These features define (a part of) the basic usage control scheme. These works also discuss the architectural support for usage control enforcement. Nevertheless, the $UCON_{ABC}$ model is a one-to-one policy model and doesn't give a natural support for the situation of assets aggregation. Moreover, the enforcement architecture doesn't present the detail mechanism for inspecting low-level 'usage' activities on consumer platform.

The research of Docomo Euro Lab [131] [130] [247] [248] [29] [249] proposes a taxonomy for the 'usage control' policy related factors [130], e.g.

usage activities, context attributes, etc. The mechanisms for inspecting low-level 'usage' activities at consumer platform, focusing on system calls, are also discussed [129] based on a holistic point of view. Policy breach is defined as the status leading to information transference to unallowed 'descriptors' (representing process, file on the disk, socket, memory area, etc) in the system. A policy refinement method mapping high-level rights definition to low-level constrains is also given, bridging policy and enforcement. Nonetheless, their policy model doesn't give a natural support for collaborative context.

This leads us to defining a collaborative usage control policy scheme, using 'policy aggregation' and 'context management' methods to deal with the collaborative contexts. As far as enforcement architecture is concerned, a slight extension is needed to generalize their 'system status' view to other platform as VM, Web, etc. Implementation architecture can be set either relying on TPM technology, for consumer platform integrity measurement and trusted information storage [134], or use a GRID-based implementation as in [66]. In this last case, policy language is based on POLPA [201] and the prototype authorization engine is implemented using Globus Toolkit [102].

These works pioneer the 'usage control' research. Nonetheless, they are proposed for traditional scenarios, whereas collaborative context requires an 'upstream provider control'. As asset can be merged during the collaborative business process, the providers' policies should take co-effect upon the merged asset, in order to protect providers' intellectual properties in the asset. Therefore, from a 'collaborative context' point of view, an access / usage control model should be able to aggregate policies according to assets derivations. This involves a 'context management' component that is able to analyze the assets derivation pattern, as well as a policy integration mechanism that ensures the resulting policy reflects the original goal of providers, based on detecting potential conflicts among their original policies. Besides, a vocabulary base including security factors in collaborative context is needed to facilitate users' policy authoring works. Such vocabulary bases should be recognized by all the partners in a context, in order to achieve interoperability among their policies.

Some recent works extend the usage control model to deal with policy aggregation:

- 'xfACL' [215] is a policy scheme which combines XACML and RBAC, highlighting issues of 'attribute representation' and 'decision aggregation'. It possesses a PDP with Microsoft 'F sharp 2010' and '.NET 4'

platform.

- [265] is a 'federated rights expression model', in which content providers, identity providers and (consumption) action providers can be distinct parties. Each party has its own policies defining in what condition their resources (content, credential, rendering capability) can be used by the partners. It enables an open DRM policy framework but, as in the previous work, enforcement mechanism isn't presented.
- Distributed context based on 'Share Date Space (SDS)' is presented in [259] [258]. It directly adopts many features from $UCON_{ABC}$.
- [268] designs a protocol, HTTPA, to enhance the HTTP protocol by requiring data consumer and provider to come to an agreement before any HTTP transaction takes place. Mechanisms allowing consumers to express their purpose of usage and providers to express their usage restrictions are provided by the protocol.
- [20] introduces a *super-sticky release policy* model for information declassification in dissemination systems. A *release policy* defines the condition for declassifying sensitive information. When information is aggregated to create new information, the *super-sticky* strategy derives the release policy of the new information from the release policies of the original information and the local release constraints imposed by the creator of the aggregated information.

There are also works that relate access/usage control with jurisdiction clauses. For example, [127] proposes a *data-purpose* algebra and uses it to model part of a Privacy ACT. The purpose is to build a policy aggregation and derivation framework, implemented in the governmental information fusion center, that accommodates legislation factors through Semantic Web technology.

However, these works seem to be in early stages and didn't give detail discussion about their features for handling collaborative context. For example, some of them only propose the policy aggregation mechanism (as [20]), or only proposes the context management mechanism ([268] [265]). Another work ([215]) has both mechanisms but doesn't provide a vocabulary base. Whereas others ([259] [258]) possess vocabulary base but don't discuss any context management method in detail.

As far as implementation is concerned, only a few of them have built the engines to support policy aggregation process [66] [215]. Even fewer has proposed policy enforcement architecture [268].

Aiming at providing a comprehensive solution for end-to-end security management in collaborative context, we take usage control policy as foundation and extend it with methods for policy aggregation and context management, as well as defining a vocabulary base collecting security factors in collaborative context, enabling interoperability between policies and facilitating policy authoring works.

2.5 Conclusion

Our purpose is to build an end-to-end security management system based on a usage control policy model comprising security factors from both 'intra-' and 'inter-' organizational levels (chapter 3). The policy model is implemented thanks to XACML language and associated with a vocabulary base (see chapter 4). We also design a policy aggregation mechanism and collaboration context management mechanism (chapter 5). The enforcement architecture is lastly developed and the performances of several components are tested (chapter 6). These works should make our system one of the most comprehensive collaborative usage control system. Nonetheless, several limits exist, making our system far from complete, as discussed in the conclusion chapter (see chapter 7, future works are also identified).

Chapter 3

Applying DRM into Enterprise information system

This chapter analyzes the new risk that rises with federated information system, led by the trend of business federation and collaborative enterprise. Based on this, a solution for end-to-end protection of corporate patrimonial value during its full lifecycle is described. Our approach relies on a collaborative-context oriented policy model that coordinates security requirements from multiple partners. Its implementation includes components for policy based security management and usage control in collaborative context.

3.1 New risk in new paradigm

Inter-organizational business federation is the leading direction of economy growth. With the development of knowledge economy and service economy, multiple organizations open their information system, integrate their business processes to form a virtual 'collaborative enterprise'. Such a collaborative paradigm has reveal advantages: it improves information communication and knowledge sharing, reduces cost and enhances business agility. Information technology fits to the requirement of this decentralized and agile paradigm.

On the other hand, the SOA and Web Service methodology expose corporate patrimonial value as services, facilitating the dynamic cooperation among organizations. It comes up with concrete implementation practices – e.g. ESB – that enable smooth exchanges of information. With the support

of SOA and Web Service, business federation can be built more easily by setting shared business processes and Information Systems.

At the same time, security concern impacts the decision of sharing information and knowledge. As such decisions are made on-the-fly in the business federation process, they require more prudence. Partners in a business federation worries not only about threats (from outside) and vulnerabilities introduced by the complex interconnection of information systems, but mostly on the way to ensure the protection of their assets while releasing them to partners ("consumers", see figure 3.1). The new risks of contract bleach and misuse of intellectual property lead to the rise of security requirements. The surveys made by IBM [189], Gartner [143], MIT [149] and European Network and Information Security Agency (ENISA) [78] show that risks as 'handling over sensitive data to a third party', 'loss of governance' and 'threat of data breach or loss' are major barriers for moving to collaborative paradigm (such as SOA and Cloud Computing). Such risks bring security to the end-to-end scale: to ensure the protection of corporate asset value during its full lifecycle (and not only during the exchange stage). This involves protecting it from un-allowed usage by partners, as well as from external threats.

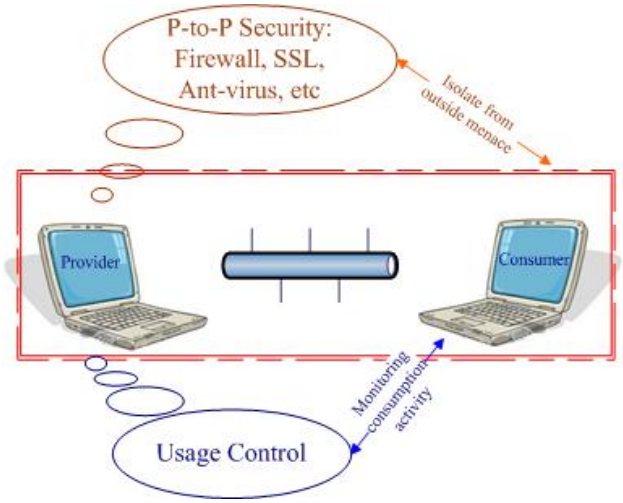


Figure 3.1: End-to-end security

Such an end-to-end security requirement roots from the organizational goal of protecting intellectual property and trade secret. During a long period, many researches result in enhancing intellectual property protection

with improved juristic practices available but few technical solutions from IT perspective.

In a digital world, the Digital Rights Management (DRM) technology fits the 'usage control' of intellectual property. DRM allows a content provider to define policies for regulating its assets consumption activities. It also offers monitoring means to enforce the policy, thanks to software components on the consumer side (usually a specific "player"). By this way, the digital content never leaves the direct control of the provider. The primary DRM technology does not suit the inter-organization business federation context, where data may be out of the direct control of its provider when it is released to (and consumed by) client side information system. However the DRM philosophy can be borrowed to the business federation context.

Some IT research and practices in federation context aim at improving the trust relationship between partners, promoting the motivation to exchange information and value. One of the most influential approach is 'trust' assessment, which evaluates the trust-worthiness of a participant, based on feedbacks on its past behavior provided by multiple partners. Trust research aims at choosing partners with better secured systems and more reliable behavior patterns. It is a viable direction toward protecting corporate patrimonial value in an indirect way. More recently, some works are achieved to support intellectual property protection in collaborative IS, under the research of 'usage control', 'privacy' or 'information flow control'.

Achieving such protection goal requires several features for the federated information system:

- Participants should be able to express their requirements with a policy model. The policy accommodates the security factors and the usage control (consumption activity) factors. It supports the coordination and aggregation of requirements from multiple providers, as the artifact in a business federation is usually co-produced by more than one provider.
- The partners' security attributes must be visible to each other, and a mechanism to support the security-oriented negotiation based on the attributes and policies must be set.
- The enforcement of policy relies on *trusted third party* and imposes a monitoring module on the participants' systems. This module inspects the consumer actions upon the provider assets and compares them with the actions defined in provider's policy. If un-allowed actions take place,

the monitoring module reports to the assets provider, and the event is logged. Such a mechanism controls the consumer system activities and confines its operation upon provider assets within the regulation of policy. This monitoring mechanism should be distributed in order to fit the dynamicity of business federation. It should also respect the privacy of partner's Information System. This is why a trusted third party is introduced protect intellectual property and trade secret, as a party usually would not allow other peers to look into its own information system.

Our approach is based on the basic thoughts in DRM and technical progresses in several fields, e.g. Semantic Web, Policy, Workflow etc, in order to enable an end-to-end security that protects assets during their full lifecycle in the setting of collaborative context.

3.2 General architecture

End-to-end security in collaborative context depends on several key elements, such as partner identification, access control policy model that expresses partners' requirements, collection of partners' attributes to support policy decision, monitoring of partner behavior, etc. This section summarizes these key elements and describes the general design for providing end-to-end security management capability to collaborative information system.

3.2.1 Target

A business federation uses shared business processes designed as a collaborative workflow where multiple organizations open their information system, expose functionalities as services and cooperate to achieve a common business goal. In the federation, partners have contractual links to ensure the quality of service. To protect their assets, partners are engaged according to a Security Service Level Agreement (***SSLA***) which is a set of regulations defining the security aspect of the collaboration context. Services consumers must respect the ***SSLA***.

Such peer-to-peer security negotiation (and configuration) approach involved by the SSLA fits the inter-organizational information system, as a central authority can not impose the security configuration of such distributed organizations.

When asset providers' require an end-to-end protection of their corporate patrimonial value, they require not only secured communication channel and secured information system on the consumer side (to mitigate security threats from outside), but also that the consumer uses their assets (and the related intellectual properties) in an allowed way. We propose a fine-grained access control policy model enriched with DRM elements to express such requirements. Our policy model incorporates factors for secured communication channel and secured platform, as well as allowing providers to define authorized consumption activities upon assets. Such a policy model provides the basic specification for end-to-end protection of a provider's resource. The provider policy defines the 'Requirement of Protection' (RoP) on its assets, whereas the consumer policy declares the 'Quality of Protection' (QoP) it has, referring to its security attributes, history activities and reputation records.

In a business federation, all the partners must share assets protection requirements. Instead of maintaining separate ***SSLA*** between each provider / consumer pair, we analyze the nature of a business federation and design a Collaboration-context oriented ***SSLA*** management system. In a business federation process, providers' assets are usually merged and mixed to produce a final artifact, e.g. the asset of the business federation. This involves that providers' policies are aggregated into the ***SSLA***, using an aggregation process launched after the negotiation process between providers and consumers. This aggregation process requires mechanisms to detect potential conflicts between partners' policies and decide whether these partners can work together or not. For example if two partners use incompatible communication mechanisms, they can not work together.

The policy implementation requires a policy engine to facilitate the policy negotiation between providers and consumers and an engine to aggregate partners' policies. The aggregation engine selects the policies that do not conflict with each other. It combines them to set the ***SSLA***. Service consumer has only to follow the ***SSLA*** to fulfill end-to-end security requirements of the providers.

This involves that we only need to aggregate the policies of the providers whose assets are aggregated. Consequently, a 'context manager' functionality is required to decide which policies should be aggregated.

As for policy enforcement, the central task is to monitor the consumption of assets. Our approach relies on an intrusive components provided by a trusted third party that is plugged into the consumer system. It monitors

the assets exchanged between partners (service access or information communication) and the consumer side operations upon the granted assets. If a consumer achieves activities that are not allowed by the provider policy, the enforcement mechanism will halt the business process and report alarms to provider.

Other services are also required to implement our policy-based model, such as identification service, log service, policy repository manager, etc.

To fulfill this end to end protection, our work is organized according to the following steps:

- First, an architecture is designed to support end-to-end security management in collaborative context. Necessary components and their functionalities are described in this chapter, including a global description of the sample use case and the implementation approach.
- Second, a policy model adapted to the collaborative scenario associated to business federation process is designed fitting the analysis of end-to-end security requirements in such federated context.
- Third, the service supporting policy negotiation, policy aggregation and context management is defined.
- Fourth, the monitoring mechanism for the enforcement of policy is discussed. It is low-level and fine-grained, allowing the definition and inspection of operation upon asset (information) occurred in consumer platform, as well as secured data storage and data exchange.

3.2.2 General architecture design

Based on observations of federated business process, we extract some important issues that impact our work:

Collaboration context

A collaboration context is an aggregation of services (representing partners' capability) in the collaborative ecosystem. The enactment, evolution and termination of such ecosystems are coordinated through peer-like activities as negotiations, under the regulation of some policies, in order to implement QoS control, conventional security services or usage control.

Identification of entities

For the end to end service/information usage control, a basic requirement is to identify entities globally. In the peer-like ecosystem, entities

can choose more than one authority to register their information. This leads to issues on how the authorities recognize, understand believe in each other.

Policy expressiveness

The policy should express the asset access condition. Security factors, audit trail records, current states concerning the involved business process can be taken into account. The policy should also express the allowed operations upon the asset.

Policy negotiation and aggregation

The policy model should be collaboration oriented, supporting the coalition of participants policies while resolving potential conflicts. It includes referencing entities' policies and reasoning about whether it fits the service aggregation requirements in a collaboration context or not.

Dynamic aggregation and change management

Due to dynamic business process composition and modification (such as the dynamic replacement of services in web service context), the issue of dynamic context changes management has a great impact on the security management. Policies should express the current state of the collaboration context and entail participants' requirements related to security.

Enforcement

The effectiveness of enforcement is critical for the reliability of the policy system. End-to-end policy model demands an enforcement mechanism that protects the participants' assets even after sending them to client side system, as well as inspecting the consumption activities.

Knowledge base

Beside the negotiation and enforcement mechanism, end-to-end policy model also requires supporting mechanisms such as policy repository, participants' attributes management and transaction events logging. These components should be consistently incorporated into a service infrastructure.

The architecture we propose to fit these requirements is built to facilitate the security management thanks to policies in a collaboration context. The requested functionalities are provided by components exposed as services that cooperate with each other (as shown in figure 3.2):

Our architecture includes:

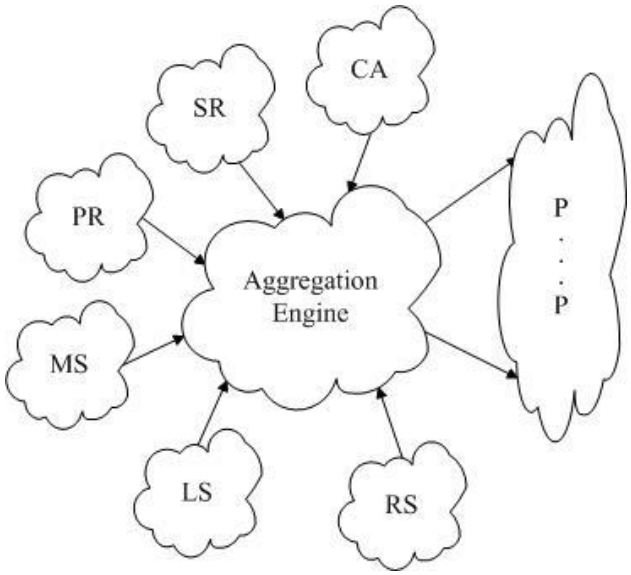


Figure 3.2: Conceptual architecture

Certificate Authority (CA)

A trusted third party serves as an authority for certifying the identities of the entities. In the collaborative ecosystem there may be more than one authority. To support large scale business federation, there should be mechanisms for the authorities to certify each other, in order to provide a federated-identity base for partners to 'recognize' each other. The authority hierarchy can be a solution as one authority issues certificates for others.

Service Registration (SR)

As discussed in SOA architecture, Service Registration is a service that maintains the information on available services. The services providers register their services information so that they can be searched and retrieved by service consumers. The service information is described using a service description language, defining basic attributes, as access address, message format, etc. It also provides index to their QoS-related attribute records in the Attribute Repository.

Attribute Repository (AR)

It is used to maintain attributes related to Quality of Service (QoS). Security-related attributes, as one facet of Qos attributes, are the main

concern in our system. Such a design increases the trust level upon the information provided to partners.

Policy Registration (PR)

It is used to maintain the services' policies, including the end-to-end security policies (which is the main concern of our work), and possibly other 'QoS' policies (which leads to a more general perspective of collaborative context management). The mechanism used is similar to the service registration. At the implementation step, functionalities of the service registration and the policy registration may be merged to provide a comprehensive service registration.

Aggregation Engine (AE)

It includes three functionalities:

- The 'policy negotiation engine' is used to decide whether two parties can work together, one acting as a provider, the other as a consumer. The decision is based on their policies and security attributes. Typical examples are XACML negotiation engines as 'SUN XACML implementation' [287] and 'JBOSS XACML engine' [67]. The 'policy negotiation engine' is the basic functionality of the Aggregation Engine (AE).
- The 'policy aggregation engine' is a major component for Collaborative Usage Control. In a collaboration context, multiple providers co-produce the final artifact. During this process, assets of different providers are merged. The 'policy aggregation engine' merges the policies of the providers accordingly. The security profiles of the consumers are then checked according to the merged policy, by the negotiation engine, to ensure that the different policies of the providers of the (final or mid-way) artifact are respected by the consumers. This process is based on the 'Collaborative Usage Control Scheme', which is discussed later in this chapter.
- The 'context manager' determines which policies should be aggregated. It implements a kind of 'down-stream information control' which ensures that a provider's policy upon an asset remains respected, even after the asset is merged into another artifact produced according to the context. Thus it needs to recognize the pattern of assets derivation. This is done by analyzing the call-response procedures among the partners. Such process can be

defined by a script language, e.g. WS-BPEL. As defined in SOA reference architecture, 'Orchestration' is a technique used to compose hierarchical and self-contained service-oriented business processes that are executed and coordinated by a single agent acting in a "conductor".

The partner call-response procedure can also be extracted from message exchange patterns. For example, 'Choreography' is a technique used to characterize and to compose service-oriented business collaborations based on ordered message exchanged between peer entities in order to achieve a common business goal. Briefly, the 'context manager' mechanism is similar to the 'information flow control' technology. It manages the asset propagation during collaboration. Thus it also copes with the dynamic aggregation and change management, including the evolution and termination of the context.

Monitoring Service (MS)

The monitoring service carries out the task of inspecting service interaction context. For example it checks the records of execution log, to support regulation enforcement. It uses some auditing metrics to examine whether, and to what degree, the regulations are complied with. A particular concern of the information given out by a service provider is the 'continuous right management' which means ensuring that the information will not be misused by any participants after exchanging it.

Logging service (LS)

Execution log is a major mean for activities inspection while running a business process. Every participants and coordinators in a process generate logs about their actions and the messages they exchange with others. It can be more efficient and effective to maintain a centralized repository of all the logs, which serve as 'provenance' of the context and provide 'proof' information of security policy enforcement, e.g. whether the policy is breached by a consumer, when and how (by which operations).

Reputation Service (RS)

Reputation service is an operational component used to store the evaluation results of the monitoring service. It supports reputation-based service selection. By this way, each participant chooses a reputation service to register its reputation records. This record may be qualita-

tive or quantitative, depending on the scale of the auditing metrics.

These key components provide the necessary functionalities for the dynamic aggregation of collaboration context in the collaborative ecosystem. Each component presents its capability as an atomic service. For example the 'Policy Registration Center' provides a portal for any participant of collaborative business to deposit it's policy assertions, in order to be referenced by others. They can also be combined to set a composite service. For example when a collaboration context needs to replace one of its participants, the Aggregation Engine receives the request, searches the Service Registration Center (SR) for services which fit the functional requirements, retrieves their policy assertions from Policy Registration Center (PR) and their reputation attributes from the Reputation Service(RS). Then the aggregation engine uses a reasoning algorithm to select "the best" service for the request and performs policy negotiation for including this service to join the collaboration context. It also retrieves auditing metrics and provides them to the Monitoring Service (MS), which inspects the interaction in the collaboration context.

3.2.3 Global organisation

Our approach is based on a policy model that expresses partners' end-to-end security requirements in the collaborative context. The components of architecture serve to manage policy negotiate among partners. By this way, the security requirements of multiple partners are coordinated and fulfilled.

We use a simple use case (see figure 3.3) of Web Service composition to improve the following discussions. It consists in some sample policies presented in an abstract syntax (which is described in detail in chapter 4).

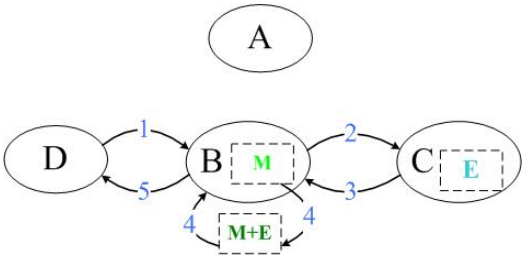


Figure 3.3: A composite service use case

The medical information of Alice (A) is kept by 'Bonetat clinique'(B). Part of the information- 'cardiac exam'- is taken from a medical examination laboratory 'Cardis health'(C). An assurance association 'Deirect assure'(D) consults A's medical information from B. In order to reply D, B contacts C to get the 'cardiac exam' of A.

These partners are using Web Service for information providing and acquisition. The sample business process includes the following steps:

(1) D contacts B, requiring M ('MedicalInfo of A');

(2) B contacts C, requiring E ('Cardiac Exam of A'); if B is allowed to get E,

(3) C sends E to B;

(4) B merges E into M; if success (involving a policy aggregation and conflicts detection process),

(5) B answers to D.

The policies of atomic services B, C are as the following:

RoP_B: (1) Anyone who wants to read A's medical information and comprise it with other data must have A's certification; (2) It must also have accessed the information within recent 90 days (active partner); (3) Information should always be encrypted with 'RSA'; (4) Those directly reading the information from B should use 'SSL' as delivery channel.

RoP_C: (1) Anyone who wants to read the information that consists in A's cardiac exam result and to use it with other data, must have A's certification; (2) It must also have accessed the information within recent 10 days; (3) Those who read (get a copy of) the information should delete it within 30 days.

Suppose that all participants have A's certification; B has accessed A's cardiac exam result from C in recent 9 days; B always deletes the cardiac exam within 20 days after having read it; D has security communication channel with IPSec VPN. So we want to answer some questions: How is their requirements and attributes expressed? Will D get the 'MedicalInfo of A'?

Each collaboration context is managed by a single aggregation engine whereas more than one instance of the other components can be used in a single collaboration. For example, different parties can choose different Certificate Authorities ('CA') to implement the identity service. Trust among CAs can be established by federation mechanisms in an open environment. Moreover, a service provider can choose one SR among the many available to register its information. Each provider also selects an AR, PR and RS for the

storage of its attributes, security policy and reputation records respectively. The chosen AR, PR and RS will be indexed into SR, as registering information. A consumer searches all the SRs for the service it needs according to their functional characteristics. The registration records of the services that satisfy the functional requirements are sent to the Aggregation Engine (AE) appointed by the consumer. The consumer service (provided that consumer's functionalities is also exposed as Web Service, which is likely the case in collaborative context, e.g. SOA, GRID, etc) registration record is also sent to AE. The AE parses the registration records and retrieves information of consumer and providers, including their attributes, reputation records, history activities and security policies. The AE performs then a negotiation between the consumer and providers policies. If more than one provider-consumer pair matches according to both functional offers and security policies (and also, possibly, QoS policies), there will be an optimization process to choose the 'best' provider (according to security attribute or QoS attributes). If composite services are needed (e.g. a provider need to request the service of another provider in order to fulfill the consumer's functional requirements), the providers in the provider-consumer pairs will search SRs to further select providers. Thus the negotiation and evaluation process becomes a multipath optimization problem.

After setting provider(s)-consumer pair, the collaboration initialization process starts. Participants' security policies are integrated in the ***SSLA***, forming a set of 'Collaborative context Security Policy' (CSP) that represents the security profile of the whole collaborative context. In order to protect the outcomes of collaboration work, it should be composed in a way that the security requirements of all the providers who contribute to an outcome must be represented. Following a 'downstream information control' principle, the policies of the providers whose assets merge together are aggregated. It results in a set of RoP_{CSP}s. In other words, in a complex context (e.g. a business process defined with WS-BPEL), we can find several '*paths*' of asset ('information' or 'service') flow (and fusion). Each path deals with a ROP_{CSP} that represents the co-effect of the RoPs from the providers. As the collaboration context evolves when participants join and quit, the new providers are allocated to the different *paths*, with their 'RoP' aggregated to the due ROP_{CSP} (for an example, see the partners 'B' and 'C' in figure 3.3, and relating according discussions).

In a similar way, the QoPs of the consumers in the collaboration context is aggregated to a set of QoP_{CSP}s, which represents the Quality of protection

offered by the context to any newly joined providers. More specifically, the consumers who will access the same asset should have their QoPs aggregated into the same QoP_{CSP} (for example, as 'D' and 'B' in figure 3.3 both access C's asset, QoP_B and QoP_D can be aggregated). The aggregation mechanism ensures that there is no conflict between the policies. Such a model fits to the ad-hoc collaboration paradigm and underpins the building and enforcing of end-to-end security policy.

The ***SSLA*** is implemented by AE after the negotiation phase. It chooses a Monitoring Service (MS) to inspect the policy enforcement and a Logging Service (LS) to record 'provenance' information about activities and outcomes of the collaboration. During the business federation, the MS generates 'provenance' data and deposits them in the LS. The reputation records of every participant are evaluated and preserved by RSs. The AE maintains the ***SSLA*** and also performs negotiation between the ***SSLA*** and the policy of parties who want to join the collaboration context. If a requesting party is currently involved in another collaboration, the two AEs (one for each collaboration) must coordinate to decide whether it can join this collaboration or not, according to issues as whether its resource is occupied by the other collaboration under the 'exclusive mode', etc. If a participant requests to quit the collaboration context, the AE checks if it has released all resources it acquired according to the ***SSLA***. After the last participant quits, the collaboration context is terminated.

It is a norm in modern information system to express security requirements in policies. The unique trait of above use case is that the information provided to D comes from two providers B and C. Both of them want that their policies cover the full lifecycle of the information they provide. A collaborative-context oriented policy model should support such trait. In the following chapter we discuss the collaborative usage control model and provide more analysis on the sample use case. The analysis gives intuitive illustration on the abstract policies language, negotiation process and aggregation process in our model.

3.3 Conclusion

End-to-end security for collaborative could be achieved using a comprehensive 'usage control' policy model. It accommodates the elements that affect (and even dominate) the decision of sharing assets between partners.

In addition, the model can cope with arbitrary collaboration patterns, from static (but layered) application, as GRID or Cloud Computing, to dynamic complex business process, as processes defined by WS-BPEL. The enforcement of such policy requires several functional components. Although the technical approaches depend on different application domain, some common functionalities can be extracted, as:

- a fine-grained policy model accommodating conventional security and 'usage control';
- a policy aggregation mechanism and context management mechanism, in order to track assets derivation and maintain policies pertaining to each asset;
- a policy enforcement architecture, including mechanism for low-level 'usage' monitoring.

In the following chapters, we introduce the detail design of such policy model and the enforcement mechanism.

Chapter 4

Towards 'Collaborative Usage Control'

Information system security management requests taking different factors into consideration, from human or legal levels to the technology level. As examples, the certification technology and trust/reputation assessment technologies take over human work while establishing cooperation, whereas watermarking technology makes forensic evidence available at the digital content.

Among these technical solutions, DRM is a representative one, with very riche features. Many efforts are made for adopting the idea of DRM in general information systems. Although some 'usage control' models have been proposed, many issues remain opened:

- First, existing models are all stand-alone models. Even though some works discuss about using these models in collaborative context, the model itself does not integrate the collaborative context from its early beginning.
- Second, no comprehensive taxonomy of the factors that should be expressed in policies has been done. Such taxonomy should lead to a vocabulary that users can use to build efficiently their security systems.
- Third, no detail discussion about end-to-end protection for intellectual property has been brought forward, in the perspective of Information Technology.
- Further, there are very few usage control policy enforcement solutions

that are comprehensive or even pertinent.

To overcome these limits, we propose a Collaborative Usage Control model, based on some renowned works as XACML [224] [166], BPEL [145], UCON policies [238] [131] [130], policy analysis technologies [185] [218] [252], etc. In this chapter, we define the basic policy model, designing the syntax for expressing briefly usage control policy and summarizing a vocabulary set. Then, semantics is given for the negotiation process, policy aggregation and collaboration context management (see chapter 5). Lastly, chapter 6 introduces the implementation of our enforcement architecture.

4.1 Collaborative usage control model

Collaborative organizations rely on contractual links between participants. This involves that a Service Security Level Agreement (SSLA) can be used between participants of a business federations. A consumer of the service (and its associated data) must follow the SSLA co-defined by service providers. Such an approach fits the inter-organizational information system requirements, as such systems lack of the central authority to manage security configuration. We propose to extend these commercial agreements to include a DRM part, associated to the information system protection. This involves defining and maintaining a policy model that integrates service and data usage control. Instead of maintaining separate SSLAs between each provider/consumer pair, we analyze the nature of a business federation and design a Collaboration-context oriented policy model.

In the following definition, we give a formal expression of the collaborative usage control scheme.

Definition 1 (Collaborative Usage Control Scheme)**.** *A collaborative usage control system is a tuple*

$$CUCON = (Sh, S, O, Ct, Rt, Ob, Rn, P, V, T, G) \tag{4.1}$$

where:

- *'Sh' (Stakeholder) is the owner of the rule, and is the owner or co_owner of the assets related to the rule.*
- *'S' (Subject) is the party that can get a Right on the asset. It is specified by a set of 'Subject Attributes' (SAT).*

- *'O' (Object) is the asset to be protected by the policy rule. It is specified by a set of 'Object Attributes' (OAT).*
- *'Ct' (Context) is the collaboration context that is associated to the status of the system infrastructure, the environment and the business federation. It is specified by the set of 'Context Attributes' (CNAT).*
- *'Rt' (Right) is the Operation upon the asset defined by 'Sh' that the Subject is allowed to achieve.*
- *'Ob' (Obligation) is the obligation that must be fulfilled by the Subject when it gets the Right.*
- *'Rn' (Restriction) is the attribute (from 'Context' set) associated to 'Rt' or 'Ob' to further confine in what circumstance they are carried out.*
- *'P' (Policy) is the usage control policy definition. It maps predicates on a set of attributes identifying 'Subject', 'Object', 'Context' to the predicates on a set of attributes identifying 'Right' or 'Obligation'.*
- *'V' (Vocabulary) is the vocabulary of attributes (and their value domains). It collects the semantic of application domain on which the authoring policy will be built. It also includes the set of Actions, for describing 'Right' and 'Obligation'.*
- *'T' (Temporal factor) 'T = {t, lc, vn}'. 't' is the temporal factors used in the attribute predicates. 'lc' defines the lifecycle of a predicate, extending the effect of the predicate to (indirect) partners corelated by the business federation. 'vn' denotes the version of the policy and is required to managed policy versioning.*
- *'G' (Aggregation algebra) is the algebra used to 'combine' the individual policies from each 'Sakeholder'.*

By this way, a collaborative usage control system takes into consideration the attributes of the assets, of the consumers, of the information system infrastructure and of the collaboration context. It enables multiple providers to co-define the policy upon the collaborative work artifact, forming the SSLA. By setting the '*lc*' factor, a provider defines its policy/predicate that will take effect beyond *direct partner*, in other words, *during the full lifecycle of the asset even after it is merged with other assets.* Consequently, algebra should be provided to ensure that the policy aggregation is consistent with the semantics of the original policies. In the following we first describe the basic usage control scheme before introducing the extension elements requested by the

collaborative context, namely the temporal factor 'T' and the aggregation algebra 'G'.

4.2 Basic usage control policy

As an 'Attribute-based access control' model, our usage control policy model uses 'many-sorted first-order predicate logic' as a foundation. Several works have been done based on the extension of such predicate logic to describe some aspects of the basic usage control scheme:

- [326] provides an abstraction of the usage control system integrated in a policy model.
- Usage control policy is a kind of 'Attribute-based access control' model, for which [303] introduced a formal model.
- Using deduction rules, [166] gives an explicit description of the semantic of XACML, a representative language of the 'Attribute-based access control' model.
- [327] developed a logical specification of the $UCON_{ABC}$ model with an extension of Lamport's temporal logic of actions (TLA), which can describe the 'attribute update' during the policy access (that is, the access and usage of assets) process.
- [72] introduced a policy analysis framework using Abductive, Constraint Logic Programming (ACLP) and Event Calculus (EC).

Their frameworks are very expressive and allow describing conditions based on attributes of subject, object or system, conditions based on usage actions and conditions based on spontaneous system events. Thus it provides a foundation for formalizing very flexible policy scheme. Several analysis tasks are viable through their framework.

These works form the foundation of basic usage control scheme. In the following, we give a brief description of basic usage control scheme. We then describe the syntax of our basic usage control model, with E-BNF, and the semantics of usage control policy using ACLP and EC. A vocabulary base is also provided for describing variable attributes related to a complex system. Several examples are presented to illustrate our policy model. The mechanism for adapting the policy model to collaborative context is presented in next chapter.

The basic construct of basic usage control system is the 'Policy Assertion' (or 'Rule'. These two terms are used inter-changeably in literatures), which is built around three central elements: '*Rights*', '*Conditions*' and '*Obligations*' (see figure 4.1).

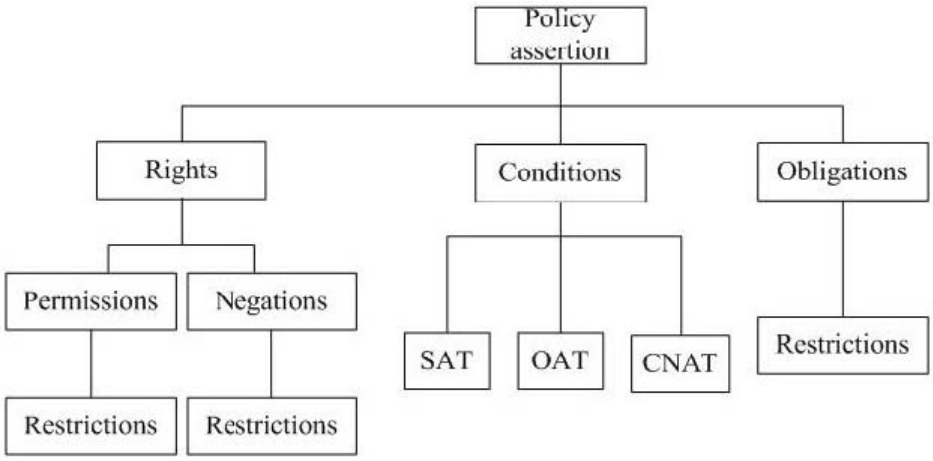


Figure 4.1: Policy assertion model

In the following we give a formal expression of the basic constructs of the policy model.

A policy ('P') is defined through the logical combination of rules ('R').

$$P ::= [\text{"}\neg\text{"}], R, \quad \{[\text{"}\wedge\text{"}|\text{"}\vee\text{"}], [\text{"}\neg\text{"}], R\} \tag{4.2}$$

A rule is a tuple

$$R = (Att, Pr, C, Rt, Ob, Rn) \tag{4.3}$$

where:

- 'Att' (Attribute) is a fixed attributes set built on the set 'V' (see 'definition 1) which is used to describe 'Rights', 'Obligations' and 'Conditions'.
- 'Pr' (Predicate) is a fixed set of predicates built on the attributes 'Att'. It consists in an attribute name, a predicate function (that is a unary, binary or trinary relation operator) and a (range of) attribute value.

- 'C' (Condition) is the condition formed by the predicate set 'Pr'. It summarizes the requirements that must be satisfied by the Subject in order to get Rights upon the Object. 'Condition' comprise information about the 'subject', 'object', 'system infrastructure', 'environment' and 'business session'. They are categorized into the set of subject attributes (namely, SAT), object attributes (OAT) and context attributes (CNAT).
- 'Rt' (Right) is the Actions upon the asset defined by the Stakeholder 'Sh' that the Subject is allowed to exercise.
- 'Ob' (Obligation) defines what the Subject must achieve when it gets the Right.
- 'Rn' (Restriction) is attached to the right 'Rt' or the obligation 'Ob' to confine the run-time status related to their fulfillments.

Rights include both allowed and prohibited actions as well as Restrictions which confine these actions. For example, a restriction 'three times' may be used to refine the right 'rendering a piece of multi-media file'. Rights are released thanks to Conditions related to either subject attributes (SAT), object attributes (OAT) or context related attributes ($CNAT$), which include the attributes related to system infrastructure, environment and business session. An example of associating Obligations to a granted Right can be 'if *read client data* (Right) then *delete acquired data in 10 days* (Obligation)' ('*in 10 days*' is a Restriction).

According to such a policy structure, we can differentiate policy assertions into several basic types.

$$Authorization\_Rule ::= Rt, " \wedge ", Ob, " \leftarrow ", C; \tag{4.4}$$

$$Obligation\_Rule ::= Ob, " \leftarrow ", C; \tag{4.5}$$

$$Restriction\_Rule ::= Rn, " \leftarrow ", C, " \wedge ", [Rt|Ob]; \tag{4.6}$$

An 'Authorization_rule' (4.4) defines that a 'Right' can be granted under a 'Condition'.

An 'Obligation_rule' (4.5) means that an 'Obligation' will be imposed (independently, without associated 'Right') upon the target of rule (usually an 'asset consumer') under a given 'Condition'.

A 'Restriction_rule' (4.6) specifies that a 'Restriction' will be imposed following the granting of certain 'Rights' or imposing of 'Obligations'.

With the basic assertions types, a rule can be decomposed, facilitating the 'policy refinement' process which interprets high-level policies into low-level operations related to specific IT infrastructure and business context These basic assertion types also provide the flexibility for composing policies. For example, Rights can be authorized without Obligations (see formula 4.4).

4.3 Policy language structure

Some 'design patterns' can be used while building a policy model. For example in the P3P project [69], policy is used to express the 'promise' of information consumer, giving provider more information to make a decision about information sharing. In XrML [305] and ODRL [227], vocabularies are provided besides the basic syntax, giving the policy scheme ability to describe concrete systems. Seeing their benefits, we use these two 'design patterns' in our policy model.

4.3.1 'RoP' and 'QoP'

Before getting a right, the consumer's request is send to the Policy Decision Point (PDP). This request consists in a set of attributes describing the status of the consumer, the system, the environment and the business session. Sometimes, however, the consumer may also provide 'purpose of usage' information or exhibit 'obligation limits' that claim the range of 'obligations' it is able to perform. For example, if a consumer must keep a piece of information for 10 hours to finish its work, it claims 'OB=Delete(Info, 10 Hs)'. By this way, the PDP ensures that the consumer is not given a right with attached obligations that it won't be able to fulfill. An example of such approach is the 'P3P' policy that declares the web site's protection of the user's private information.

In our usage control scheme, we propose to use a 'Quality of Protection' (QoP) policy to express a consumer's predefined promises about the protection it offers to the assets it demands instead of specifying attributes set in request. Accordingly, the policy that specifies the asset provider's requirements is denoted as 'Requirements of Protection' (RoP). Both (RoP) and (QoP) have a similar syntactic structure.

In a business federation, a party is an 'Asset Provider' if it releases assets to partners. As soon as a party receives assets from others, it becomes an 'Asset Consumer'. We differentiate the concept of 'Asset Provider' and 'Service Provider', as well as the concept of 'Asset Consumer' and 'Service Consumer'. A party can act as both asset provider and asset consumer in a business session. A 'Service Provider' is both an 'Asset Provider' that provides a 'service' and an 'Asset Consumer' that consumes client's 'information'. In a symmetric way, the 'Service Consumer', is both an 'Asset Provider' that provides client's 'information' and an 'Asset Consumer' that consumes a 'service'. Therefore each party policy consists in two parts, RoP_P and QoP_P (see figure 4.2).

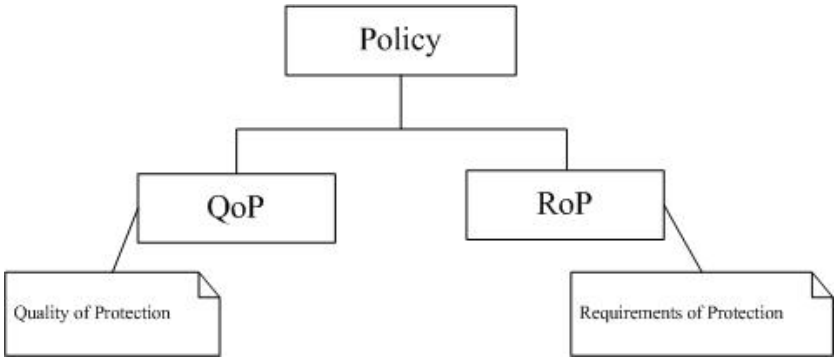


Figure 4.2: 'RoP' and 'QoP'

4.3.2 'Grammar' and 'vocabulary'

A policy scheme commonly consists in the language expression scheme, to capture the syntax and semantics of the language, and the data dictionary scheme, to represent the knowledge related to the application domain (see figure 4.3).

The language expression scheme defines the 'grammar' of the policy language. It is used to express the policy logic, such as the combination (or choice) of conditions and effects of rights and obligations. The data dictionary scheme offers a 'vocabulary' set to describe the objects and properties in real word that should be regulated in policy, such as the set of attributes of participant or the 'allowed actions' included in the rights.

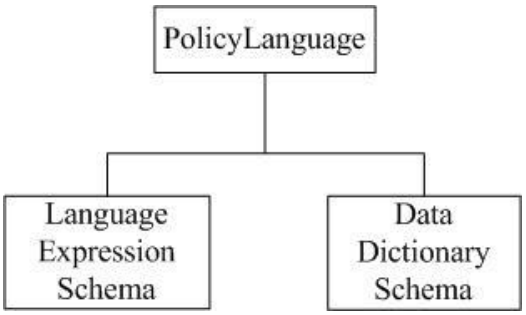


Figure 4.3: policy scheme structure

4.4 Abstract syntax

This section defines the abstract level syntax of our policy model, using a version of Extended Backus-Naur Form (EBNF) defined by ISO/IEC 14977. Table (4.1) gives the meaning of its symbols.

	Symbol	Meaning
1	commas (',')	it's used to separate a sequence of expression
2	vertical bars ('|')	it's used to separate alternatives
3	squared bracket ('[]')	expressions that can occur zero or one time are represented in it
4	curly braces: ('{}')	expressions that can occur zero or many times are represented through it
5	quotation marks ('"')	terminals are enclosed in it
6	semicolon (';')	it denotes the end of a formulae

Table 4.1: Symbols in EBNF

Sample use cases are presented, at the end of this chapter, illustrated the employment of the usage control policy in different scenarios.

Our policy model design is based on investigation of influential industrial works such as ODRL [227], XrML [305], [224]XACML, xbXML and academic works as the $UCON_{ABC}$ model [238] and the Usage control policy in DoCoMo Euro Lab [130]. Our policy model differs from former approaches in:

- A detailed taxonomy of relevant factors (based on analysis of former canonical works as BPEL, XACML, DRM, etc.) helps to define a

'vocabulary base', which users can extend and apply to their application domain.

- The policy model is collaboration-oriented. The Right of asset provider is protected during the full asset lifecycle.

In what follows, we'll use acronyms defined in table (4.2).

	Acronyms	**stands for**
1	'Sh'	stakeholder (see definition 1 in section 4.1)
2	'C'	condition
3	'SAT'	subject attributes
4	'OAT'	object attributes
5	'$CNAT$'	context attributes
6	'Rt'	right
7	'Rn'	restriction
8	'Ob'	obligation
9	'lc'	lifecycle (of a rule or a predicate)

Table 4.2: Symbols in EBNF

A policy assertion is defined upon several factors as Stakeholder (Sh), Subject (S), Object (O), Context (CN), Right (Rt), Obligation (Ob) and Temporal factor (T). A sample policy assertion looks like:

$$
\begin{aligned}
&RoP_x.(lc =' 30days') : \\
&\quad Rt(actionID = play) \\
&\quad \leftarrow \\
&\qquad Sh(A = 50; C = 30; B = 20) \\
&\qquad \wedge OAT(ID = M \wedge encrypted = Yes) \\
&\qquad \wedge SAT(partnership(organization(S), C) \\
&\qquad\qquad \wedge pay = (100Euros)) \\
&\qquad\qquad \wedge (role(S) =' tourist\_guide' \\
&\qquad\qquad\quad \vee descendentRole(S) =' tourist\_guide')) \\
&\qquad \wedge CNAT(deliveryChannel = "SSL")
\end{aligned}
\tag{4.7}
$$

In this example, the Right 'R' is granted with the fulfillment of conditions defined by factors of OAT, SAT and $CNAT$. The element 'lc' is a temporal

factor that denotes the life cycle of the assertion. 'Sh' denotes the owner(s) of the policy. We call the outcome of collaborative business process a 'C-Asset'(Collaboration Asset) and the original asset each provider offers to the collaboration a 'O-Asset' (Original Asset).

A stakeholder (owner or co-owner of a policy) is defined as:

$$\begin{aligned} Sh ::= & "Sh(", entityID, [percentage], \\ & \{; entityID, [percentage]\}, ")"; \end{aligned} \tag{4.8}$$

The Owner/Co_owner (Sh) is identified by his/ their ID(s). In our policy model, a rule can be signed by more than one stakeholder, to fit the case of Collaboration-context Security Policy (CSP), where the 'C-Asset' is co_owned by more than one stakeholder.

The 'Condition' part of an assertion is formed by 'attribute predicates' on the attributes of Subject, Object and Context, with 'AND', 'OR' and 'NOT' relations. as shown in the following formulae.

$$C ::= ["\neg"], [condition], \{[" \wedge "|" \vee "], ["\neg"], condition\}; \tag{4.9}$$

$$condition ::= SAT|OAT|CNAT; \tag{4.10}$$

The 'SAT', 'OAT' and 'CNAT' in 4.10 are the sets of predicates on attributes of 'Subject', 'Object' and 'Context'.

$$\begin{aligned} SAT ::= & "SAT(", \\ & ["\neg"], \\ & subject\_attribute\_name, operator, attribute\_value, \\ & \{[" \wedge "|" \vee "], ["\neg"], \\ & subject\_attribute\_name, operator, attribute\_value\}, \\ & ")"; \end{aligned} \tag{4.11}$$

Definition 4.11 gives the detail of the SAT expression. An example is: $Organization\_name\ equal\_to\ 'xyzInc.'$, where the attribute name is '$Organization\_name$', the operator is '$equal\_to$' and the attribute value is '$xyzInc.$'. The 'operator' factor will be represent in section 4.7.2.11.

Similarly we can define the syntax for the '**Object**' and **Context** factors as:

$$\begin{aligned} OAT ::= & "OAT(", \\ & ["\neg"], \\ & object\_attribute\_name, operator, attribute\_value, \\ & \{[" \wedge "|" \vee "], ["\neg"], \\ & object\_attribute\_name, operator, attribute\_value\}, \\ & ")"; \end{aligned} \tag{4.12}$$

$$\begin{aligned} CNAT ::= & "CNAT(", \\ & ["\neg"], \\ & context\_attribute\_name, operator, attribute\_value, \\ & \{[" \wedge "|" \vee "], ["\neg"], \\ & context\_attribute\_name, operator, attribute\_value\}, \\ & ")"; \end{aligned} \tag{4.13}$$

The 'Rights' part of an assertion can involve multiple rights. Negative symbol can be used, resulting in a negation rule.

$$Rt ::= "Rt(", ["\neg"], [right], \{[" \wedge "|" \vee "], ["\neg"], right\}, ")"; \tag{4.14}$$

A granted Right can be attached with **Restriction**, which gives more confining information about the right.

$$Rn ::= "Rn(", ["\neg"], [restriction], \{[" \wedge "|" \vee "], ["\neg"], restriction\}, ")"; \tag{4.15}$$

Rights can be refrained with Obligations, which are 'actions' that must be achieved after getting the Rights.

$$Ob ::= "Ob(", ["\neg"], action, \{[" \wedge "|" \vee "], ["\neg"], action\}")"; \tag{4.16}$$

In next section, we describe the semantics of the basic usage control policy model by elaborating the evaluation mechanism between a policy and a request.

4.5 Semantics

Negotiation is based on the request generated from an Asset Consumer's ('AC' for short) QoP ('QoP_{AC}' for short) and the Asset Provider's ('AP' for short) 'RoP' ('RoP_{AP}' for short). The decision consists in denying or granting Rights and generating Obligations. Some works [166], [85] [72] form the semantic foundation for our work. For example, the basic semantics of negotiation is in accordance with the work of [166], which formulates the negotiation process for attribute-based access control model. As far as obligation predicates are concerned, we process them as other attribute predicates as proposed in [85]. The way a decision is interpreted (result of negotiation) can be formulated thanks to logic programming and Event Calculus (which are expressive for describing decentralized system status) as in [72].

In the following we first formulate the request and the policy negotiation decision. Then the negotiation process is introduced, describing the way requests and policies are matched, including the impacts of multiple rules combinator.

4.5.1 Request and decision

A request is generated from an Asset-Consumer's QoP. It declares the consumer's attribute(SAT), the attributes of the assets (OAT) it requires, the context attributes (CNAT, which includes the attributes of the infrastructure, the environment and the business session), the Rights it requires and the Obligations it will align with. In a state-based system such as a usage control system (and further, in a distributed system as collaborative usage control scheme), the temporal information of the access-related actions, as request generation, decision and policy evaluation response, enforcement of decision, etc., should all be recorded. Such information is regulated as (events) attributes of the business session.

$$\begin{aligned} req(Rt, QoP_{AC}.SAT, QoP_{AC}.OAT, \\ QoP_{AC}.CNAT, QoP_{AC}.Ob, t) \end{aligned} \tag{4.17}$$

When a request matches a rule in the policy, the effect of the rule is returned as a decision (usually 'permit' or 'deny' some rights). The decision indicates the 'subject', 'object', 'rights', 'obligation' and 'restriction'. The following formula shows the decision process. As our collaborative usage control scheme is a distributed state-based system, if the QoP of asset consumer

can fulfill the RoP of asset provider, Rights can be granted. Obligations can be imposed, and the rights may be followed by Restrictions. A temporal factor ('t') denotes rule life-cycle.

$$[\neg]Authorize(Rt, S, O, OB, Rn, t) \leftarrow QoP_{AC} \supset_{Sa} RoP_{AP} \tag{4.18}$$

where 'Rt', 'S', 'O', 'Ob', 'Rn', 't' stand for 'right', 'subject', 'object', 'obligation', 'restriction' and 'time'. The function $\supset_{Sa}$ denotes that the QoP_{AC} matches the RoP_{AP}. This involves matching a request with each rule and then considering the effect of multi-rules related with a 'combinator'.

4.5.2 Matching mechanism

[166] presentes a description of XACML negotiation process based on abductive rule. Using a similar approach, we can describe the matching process between request and policy in a general Attribute-based access control policy negotiation.

Attribute matching The following four formulae describe the matching process of an attribute name in the policy rule with attributes in the request. The formula (4.19) defines if the 'issuers' of two attributes are in one Certificate-Chain (CC), the attributes predicates are trusted between the two partners, namely the requester and the policy owner. This (4.19) is necessary in a decentralized context (which is the case of a business federation) as the partners' attributes may be issued (certificated) by different issuers: trust among these issuers is necessary for the partners to acknowledge the status of each other.

$$\frac{\exists CC : issuer_{AR} \in CC, issuer_{AQ} \in CC}{issuer_{AR}, issuer_{AQ} \models True} \tag{4.19}$$

In formula (4.19), 'AR' stands for an attribute in the rules, 'AQ' for an attribute in the request. We use the notation convention that for an element 'P', 'X_P' refers a child element (or property) of 'P'. So '$issuer_{AR}$' refers to the issuer of the 'AR'. '$\models$' stands for the 'entailment' relationship. '$A \models_L X$' means that X is provable from A for a language L. In other words, X is a semantic consequence of a set of statements A under a deductive system

that is *complete* (all valid arguments are deducible – provable) and *sound* (no invalid arguments are provable) for a language L.

The formula (4.20) asserts that if the attribute names (*attr-id*) of the attribute in the rule ('AR') and attribute in the request ('AQ') are the same, their types (that is, data type of the attribute value) are the same and they are issued by trusted issuers, the 'AR' and 'AQ' can match each other, depending on the predicate function 'fcn_M' of 'AR' and the attribute value 'AV' provided by 'AQ', in other words, if the attribute value provided by the request is in the attribute value range defined by the rule.

$$\frac{\begin{array}{l} attr\text{-}id_{AR} = attr\text{-}id_{AQ} \\ type_{AR} = type_{AQ} \\ \forall issuer_{AR}, issuer_{AQ} : issuer_{AR}, issuer_{AQ} \models True \end{array}}{AR, AQ \models_m AV_{AQ}} \tag{4.20}$$

The formula (4.21) asserts that for an 'AR', if there is an 'AQ' in the request 'Q' matching it and if the AR and AQ are describing the same type of entities (entities include 'subject', 'object', 'context'), the 'AR' matches the 'Q'.

$$\frac{\exists AQ \in Q : cat_{AR} = cat_{AQ} \qquad AR, AQ \models AV_{AQ}}{AR, Q \models_m AV_{AQ}} \tag{4.21}$$

The formula (4.22) asserts that for an 'AR', if all the 'AQ' in 'Q' don't match it, 'AR' is not matched by 'Q' ('Indeterminate'). It means that an attributes required by the policy is not exhibited in the request. Some access control systems use a 'certificate discovery' process to find the 'missing' attribute. We also use this approach in our system.

$$\frac{\forall AQ \in Q : AR, AQ \not\models_m AV_{AQ}}{AR, Q \models_m Indeterminate} \tag{4.22}$$

Attribute predicate matching The following two formulae gives the semantic for matching an attribute predicate in a rule with a request. The syntactic element 'M' stands for the attribute predicate. The 'fcn_M' is the predicate function (see 'Pr' in formula 4.3) that operates on a 'AV' (attribute value) domain.

The formula (4.23) states that for an attribute predicate 'M', if its attribute name 'AR' is matched by an attribute name 'AQ' in the request 'Q' and the 'AV' of 'AQ' satisfies '$fcn_M(AV_M)$' (that is, in the range of value defined by the predicate function 'fcn_M' and attribute value in 'AR'), 'M' is matched by 'Q'. The formula (4.24) states the 'Indeterminate' situation, that is when the 'AR' doesn't match 'AQ' in 'Q'.

$$\frac{\begin{array}{l}\exists AR \in M : AR, Q \models_m AV_{AQ} \\ fcn_M(AV_M, AV_{AQ}) = True\end{array}}{M, Q \models True} \tag{4.23}$$

$$\frac{\exists AR \in M : AR, Q \models_m Indeterminate}{M, Q \models Indeterminate} \tag{4.24}$$

The '**predicate function**' 'fcn_M' can be (as stated in the work of Agrawal et. al [4] and Lin et. al [188]) classified into five categories:

- Category 1: 'One variable equality constraints';
- Category 2: 'One variable inequality constraints';
- Category 3: 'Real valued linear constraints';
- Category 4: 'Regular expression constraints';
- Category 5: 'Compound Boolean expression constraints'.

The names of these categories explain their meaning to some extent. Chapter 2 presented their information in detail.

The whole space that the 'constraints' can operate over is the 'AV' domain. As proposed in XACML standard [224], it includes the domains of data types of Boolean, date and time, numeric data, plain string and string with special meaning: x500Name, rfc822Name, ipAddress, dnsName and URI. The URI data type is versatile, especially with the development of recent ontology-definition technologies as OWL [199] and SWRL [133].

Ontology allows the construction of domain knowledge by defining the relation between concepts. For example in a 'network ontology', one may find statement: SSL *is subclass of* Secured channel, where 'SSL' and 'Secured channel' are two concepts representing two sets of things (usually called 'Class') and '*is subclass of*' is a relation defining that any entity is a 'Secured channel' if it is 'SSL'. Ontology-based technologies are more and more frequently used in defining domain knowledge.

As far as policy authoring is concerned, an attribute domain incorporating ontology-based vocabulary has a great impact on the predicate function (thus to the attribute matching between request and policy). This leads to a new predicate function category, related to semantic relationship description:

- Category 6: Semantic relation constraint (equal to the 'object property' in protégé [132] OWL).
  $x \doteq c$, where x is a variable, c is a constant (a URI denoting a concept, in OWL: class or individual) and $\doteq$ is the user-defined 'semantic relation'.

Examples to illustrate the impact upon attribute matching can be easily found. Given the ontology information 'SSL *is subclass of* Secured channel', an attribute definition 'Network(a)='SSL' in the request matches the attribute definition 'Network(a)=Secured channel' in the policy. Besides, if the vocabulary consists in ontology information 'SSL *not compatible with* IPSec', an attribute definition 'Network(a)='IPSec' in the request leads to a "not matching" conclusion regarding the attribute definition 'Network(a)=SSL'. Of course if there are two policy authors in a system and if they want their policies to interact, they must share the same meaning on concepts and relations defined in their vocabularies.

Request matching The following three formulae give the semantic of the matching effects between a rule (R) and a request (Q):

- The formula (4.25) denotes: If all attribute predicate 'M' in a rule 'R' are matched by 'Q', the access decision is provided by the effect of the rule:'$Effect_R$'.
- The formula (4.26) states: If some 'M' in 'R' are not matched by the request 'Q' (described by formula 4.24 and 4.22), the matching between 'R' and 'Q' is 'Indeterminate'.
- Otherwise, if the matching process between 'R' and 'Q' is not in the state of 'Indeterminate' but doesn't come up with '$Effect_R$', 'R' is 'Not Applicable' to 'Q' (see formula 4.27). It refers to the situation that formula (4.23) is not satisfied: all 'AR's match with 'AQ' but some 'M's do not match 'Q' (due to the failure of satisfying '$fcn_M(AV_M, AV_{AQ}) = True$').

$$\frac{\forall M_R : M_R, Q \models True}{R, Q \models Effect_R} \tag{4.25}$$

$$\frac{\exists M_R : M_R, Q \models Indeterminate}{R, Q \models Indeterminate} \tag{4.26}$$

$$\frac{R, Q \not\models Effect_R \qquad R, Q \not\models Indeterminate}{R, Q \models NotApplicable} \tag{4.27}$$

Negation as failure The formula below 'flattens' the multi-valued decision of rule evaluation to the binary-value of 'Permit' effect and 'Deny' effect. We adopt the 'Negation as Failure' principle to set 'Close Policy' model, that is, a request is denied by default, unless the requester shows all necessary credentials to successfully match with a 'permit rules' (rule with 'permit' effect):

$$\frac{R, Q \models Indeterminate \vee R, Q \models NotApplicable}{R, Q \models Deny} \tag{4.28}$$

Obligation Obligations are deemed as constraints for future execution behavior of the system [85]. The promise of obligation can be seen as a pre-condition for granting rights: $grant(rights) \leftarrow fulfill(condition) \wedge promise(obligation)$. We use

$$QoP_{AC}.Ob \supset_{Sa} RoP_{AP}.Ob \tag{4.29}$$

to denote that the 'Obligation' of the Asset Consumer's QoP satisfies the 'Obligation' of the Asset Provider's RoP. Further, the treatment of obligation is a bit more complex than for other components in the policy, as obligations and rights can both be defined with actions. The negotiation process must also evaluate potential conflicts between the obligations in request with authorizations in the policy, precisely:

$$\frac{\begin{array}{l}\exists M_{Rt} \in Rt_R, AR \in M_{Rt}, AQ \in Ob_Q : \\ AR, Ob_Q \models_m AV_{AQ} \qquad fcn_M(AV_M, AV_{AQ}) = False\end{array}}{Rt_R, Ob_Q \models False} \tag{4.30}$$

In formula (4.30), 'Rt_R' is the 'Right' element of a rule 'R', 'Ob_Q' the 'Obligation' element in the request 'Q'. It describes that if there are an attribute predicate 'M' in 'Rt_R' and an attribute predicate 'AQ' in 'Ob_Q' that they have the same attribute name but their attribute value range 'AV' are not consistent w.r.t. the predicate function 'fcn_M', the 'Rt_R' is not consistent with 'Ob_Q' (In other words, the 'Obligation' defined by the request is not consistent with the 'Right' authorized by the rule).

4.5.3 Rule combinator

When a request is matched by multiple rules, especially both 'permit rules' and 'deny rules' (rules with 'deny' effect), a 'combinator' is required to solve the conflicts [218] [185] [45] [32]. Although conflict is a knotty question in policy administration [207] [302], it is however a featured characteristic of flexible access control scheme [224].

In our usage control model, we adopt the 'Deny-override' combinator as used in XACML, to get a stringent usage control strategy. A request that is matched by both a 'permit rule' and a 'deny rule' is denied. By giving deny rules precedence, we get a more restrictive policy model. It helps avoiding the unintentional rights granting due to the fact that a slipshod permit rule takes precedent over the well-defined rules. Moreover, this strategy can be coupled with the 'Negation as Failure' principle to build a multi-layered effect space that allows fine-grained policy authoring (as illustrated in figure 4.4).

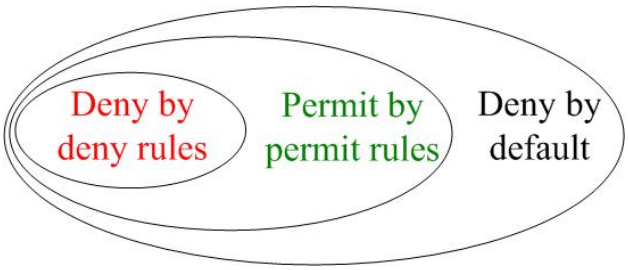


Figure 4.4: Partition of requests domain with multi-layered effects

The following formulae describe the evaluation of multiple rules coordinated by 'Rule Combining Algorithm'. The deductive rules we use are similar to the work of [166], as our basic usage control model is simpler than the XACML semantic. The later differentiates attribute predicates to the set

of 'Target' and the set of 'Condition'. It supports several 'Rule Combination Algorithm' whereas we only adopt the 'Deny-override' algorithm. This simplifies our design of the 'Aggregation Algebra' (see chapter 5), which is used to 'combine' policies from multiple partners.

In these formulae, the 'ϵ' stands for the Effect of policy (in our usage control model, the 'Deny' effect), '$\overline{\epsilon}$' the opposite effect ('Permit' effect in our model), 'P' is the policy combining several (denoted by the integer 'n' in following formulae) rules ('R').

The formula (4.31) asserts: The 'P' with ' ϵ-$Overrides$ ' Rule Combination Algorithm is evaluated to Effect 'ϵ' if among all the rules 'R' that matches 'Q', there is at least one rule 'R' with 'ϵ' Effect.

$$\frac{\exists i \in [1,n] : R_i, Q \models \epsilon \qquad (R_1, ..., R_n), Q \models True}{(P\ \epsilon\text{-}Overrides\ R_1, ..., R_n), Q \models \epsilon} \tag{4.31}$$

The formula (4.32) asserts: the 'P' with ' ϵ-$Overrides$ ' Rule Combination Algorithm is evaluated to 'Indeterminate' if among all the rules 'R's that matches 'Q', there is no 'R' with 'ϵ' Effect that is not 'Indeterminate'. At the same time, there is at least one R that is 'Indeterminate'.

$$\frac{\begin{array}{l}\forall R : R, Q \models Indeterminate \wedge Effect_R = \epsilon \\ \exists j \in [1,n] : R_j, Q \models Indeterminate \\ (R_1, ..., R_n), Q \models True\end{array}}{(P\ \epsilon\text{-}Overrides\ R_1, ..., R_n), Q \models Indeterminate} \tag{4.32}$$

The formulae (4.33) asserts: The 'P' with ' ϵ-$Overrides$ ' Rule Combination Algorithm is evaluated to the opposite Effect of 'ϵ' ('$\overline{\epsilon}$') if among all the 'R's that match 'Q', there is at least one rule 'R' with '$\overline{\epsilon}$' Effect, there is no 'R' with 'ϵ' Effect, and, at the same time, all the 'R's that match 'Q' as 'Indeterminate' must have Effect of '$\overline{\epsilon}$' (that is, not of 'ϵ').

$$\frac{\begin{array}{l}\forall R : R, Q \not\models Indeterminate \vee Effect_R = \overline{\epsilon} \\ \exists i : R_i, Q \models \overline{\epsilon} \qquad (R_1, ..., R_n), Q \models True \\ \forall j : R_j, Q \not\models \epsilon\end{array}}{(P\ \epsilon\text{-}Overrides\ R_1, ..., R_n), Q \models \overline{\epsilon}} \tag{4.33}$$

Lastly, the formula (4.34) asserts: The 'P' with ' ϵ-$Overrides$ ' Rule Combination Algorithm is evaluated to 'NotApplicable' to the 'Q', if all the

'R's matches 'Q' as 'NotApplicable'.

$$\frac{\forall R : R, Q \models NotApplicable}{(P\ \epsilon\text{-}Overrides\ R_1, ..., R_n), Q \models NotApplicable} \tag{4.34}$$

According to the 'Negation as failure' and 'Close policy' principle, we use the following semantics to 'flatten' the 'Indeterminate' and 'NotApplicable' effect to 'Deny' Effect.

$$\frac{P, Q \models Indeterminate \vee P, Q \models NotApplicable}{P, Q \models Deny} \tag{4.35}$$

4.5.4 Response

A response can either deny or grant Rights with some Obligations and Restrictions. In some situation, the asset attribute (OAT), and the context attributes can be modified.

$$[\neg]Authorize(Rt, S, OAT, CNAT, Ob, Rn, T) \tag{4.36}$$

State regulation: When the consumer carries out the granted rights, the 'usage event' (i.e. the consumption activities of protected asset) usually leads to change the system status, (as, e.g. the 'attribute update' discussed in $UCON_{ABC}$ model). Such events should be considered in policy authoring, e.g. for the sake of 'model conflict detection' (e.g. 'DSoD', see chapter 2 for more information). We collect such events as 'business session attributes', as a part of the CNAT (contextual attributes). As defined before, CNAT consists in the status information about 'infrastructure', 'environment' and 'business session'. Some attributes are 'dynamic' (e.g. 'usage event', 'business session event' or 'environment factor'), whereas others (the 'infrastructure' information) are rather 'static', i.e. they are not related to the 'usage event' or any other dynamic event occurring in the business session context. Following the approach of [72], we use 'Event Calculus' to describe the dynamic attributes of CNAT.

For an incoming request, the context attributes CNAT includes the asset consumer's state information concerning its fulfillment of the right (and associated obligations) this consumer was granted before. These context factors are related to the individual consumer, we denote them as $CNAT_{ind}$.

There are also other 'public' contextual information that are not related only to one party but rather to all the parties, concerning the state of the collaboration context as a whole. We denote this as $CNAT_{pub}$.

Thus, we have:

$$CNAT = CNAT_{ind} \cup CNAT_{pub} \tag{4.37}$$

The following formulae (4.38)-(4.42) define five types of $CNAT_{ind}$ statements.

$$permitted \quad (Rt, Sub, Obj, t) \tag{4.38}$$
$$denied \quad (Rt, Sub, Obj, t) \tag{4.39}$$
$$obliged \quad (On, Sub, Obj, t_s, t_e, t) \tag{4.40}$$
$$fulfilled \quad (Act, Sub, Obj, t_{init}, t_s, t_e, t) \tag{4.41}$$
$$violated \quad (Act, Sub, Obj, t_{init}, t) \tag{4.42}$$

Where

- 'Act' is the set of 'action' and the sets $Ob \sqsubseteq Act$, $R \sqsubseteq Act$.
- 'Sub' is the 'Subject'.
- 'Obj' is the Object(s).
- The 't' is the time point when the 'permission' (4.38), 'negation' (4.39) or 'obligation' (4.40) is imposed.
- The 't_s' and 't_e' are the start time and end time of the obligation respectively.
- The 't_{init}' is the time point when the fulfillment or violation of the obligation happenes.

It can be seen that these formulae express the system events that are regulated by the policy, i.e. the events of Rights or Obligation authorization and the fulfillment or violation of them.

Formulae (4.43)-(4.45) define three types of $CNAT_{pub}$ statements.

$$happens \quad (event) \tag{4.43}$$
$$holdsAt \quad (event) \tag{4.44}$$
$$broken \quad (event) \tag{4.45}$$

They express events that are not managed by policy, e.g. events in business context as 'partner join and quit' or 'session initiation or termination' or environment event as changes of 'location', 'time', 'temperature', etc.

[72] provides examples of using *Event Calculus* to express and manage system status evolution. The 'state regulation' is rather related to the 'policy refinement' work, where 'high level' policies are mapped to 'low level' system operations. As we aim at designing a usage control policy model for collaborative context, we only provide sufficient syntax and semantic elements allowing the expression of a 'high level' policy.

Another important issue is the 'Attribute Update'. [327] includes the expression of 'attribute update' action in their core policy model. We believe it is all as well to left 'attribute update' out of the core policy model, because the update process is 'implicit' with the authorization of policy (and the achievement of 'usage' right by consumer). In our system, the 'attribute update' process is managed by the 'policy enforcement' component. This will help keeping the core policy model concise.

Negotiation process of the sample use case By now, we can see that the generation of request from QoP_B is straightforward. With the definition of RoP_C and QoP_B (see section 4.8.1) we can decide B's request to 'cardiac exam' will be authorized by C.

As mentioned above, we designed a vocabulary base with semantic technology to represent the domain knowledge that is commonly useful in collaborative information system.

4.6 Translation to concrete syntax

The negotiation process requires a negotiation engine. Instead of developing a negotiation engine from scratch, we propose to implement our policy scheme using some existing rule languages supported by negotiation engines, e.g. XACML (It's an access control language that uses the miscellaneous 'attributes' of resource, consumer and environment to grant, or deny, the access authorization. It can also express various 'consumption' activities through 'Action' element.) or SWRL, which are built by prestigious projects. Their reliability have been tested by a user community world wide.

Basically, RoP assertions are translated as XACML ***policies***, whereas QoPs are interpreted into XACML ***request-context***. The translation process is quite straight: 'SAT' is defined by 'Subject' element (more precisely, using 'SubjectMatch', 'SubjectAttributeDesignator' and 'AttributeValue'),

'OAT' is defined by 'Resource', 'CNAT' by 'Environment', 'R' by 'Action', 'Ob' by 'Obligation'.

As XACML is originally designed for access control between one provider and one consumer, its rules only take effect between direct partners. In a collaborative context, the provider wants its resource to be protected during the full lifecycle. This requirement can be met thanks to some minor modifications (discussed in detail in chapter 5) of the original XACML syntax to enrich its semantic, extending the effect of some rules beyond direct partners.

Translation to SWRL is also straightforward. As SWRL is a general-purpose rule language based on 'Semantic Web' technology, implementing our policy model using it requires only translating an 'attribute predicate' to a SWRL predicate and translating a rule in our policy to a SWRL rule. Defining policy with SWRL is more error-prone than with XACML, as SWRL doesn't confine the structure of the rule. When authoring or reading a SWRL rule, it's not easy for the user to determine which attribute belongs to the subject, or to an object. Nevertheless, the flexibility brought by SWRL allows users to define more comprehensive policy structure that may be necessary to describe the complex collaborative context.

Next section presents the taxonomy of the syntax elements in our policy model and some factors relating to them. They form the general foundation of our security model for Collaborative System.

4.7 Vocabulary

A policy scheme usually possesses a 'vocabulary base', allowing domain/industry specific knowledge to be described. This vocabulary makes a user's work more convenient. There are several requirements for the vocabulary base, regarding the collaborative system:

- It should be extensible, so that users can add new concepts.
- It should be interoperable, in order to fit the collaboration context, where each organization may 'understand' the terms according to the others' vocabularies.
- It should be machine readable and processable in order to be adapted to large scale complex application.
- Consistency must be easily maintained on the conceptual structure of domain knowledge and relations among concepts.

Therefore, we build our 'vocabulary' as an ontology defined with OWL language. OWL ontology organizes concepts into a clear-cut hierarchical structure. It not only collects the terms that represents basic concepts, but it also defines relations between them. Consistency among these relations can be maintained with 'semantic reasoner' toolkits.

In this section, we give an overview of the factors we collect into our vocabulary, as well as their structural organization. Then, we introduce the taxonomy of the factors closely related to the definition of attributes of the parties in a collaborative context.

4.7.1 Vocabulary base structure

Figure 4.5 describes the general-level ontology structure of our vocabulary. Concepts are organized into three groups:

- '**End-to-end security**' expresses that an asset provider believes the consumer implements security means to secure its assets (and protect its intellectual property). This means that the consumer:
  - uses a reliable infrastructure;
  - is able to provide secure communication channel;
  - will behave according to regulation;
  - can be monitored.

  This fine-grained trust is more thorough. It extends the traditional trust concept with thoughts from access control and DRM. The 'Policy' defines regulations on operations upon services/assets and the due conditions. It also defines a party's statement about how it will protect the assets granted to it. 'Credential' provides identity certification to parties, thus serves as the foundation of trust and security. The 'Negotiation' of partners policies require some algorithms.
- 'Condition' and 'Right' relate to the usage control policy. We define four kinds of rights:
  - 'usage' is associated to asset consumption actions;
  - 'management' is used to express resource exchange;
  - 'assume local role' is for accommodating RBAC in collaborative environment;
  - 'delegation' stands for the 'delegation' mechanism which is a wide adopted business paradigm for collaborative process, e.g. role

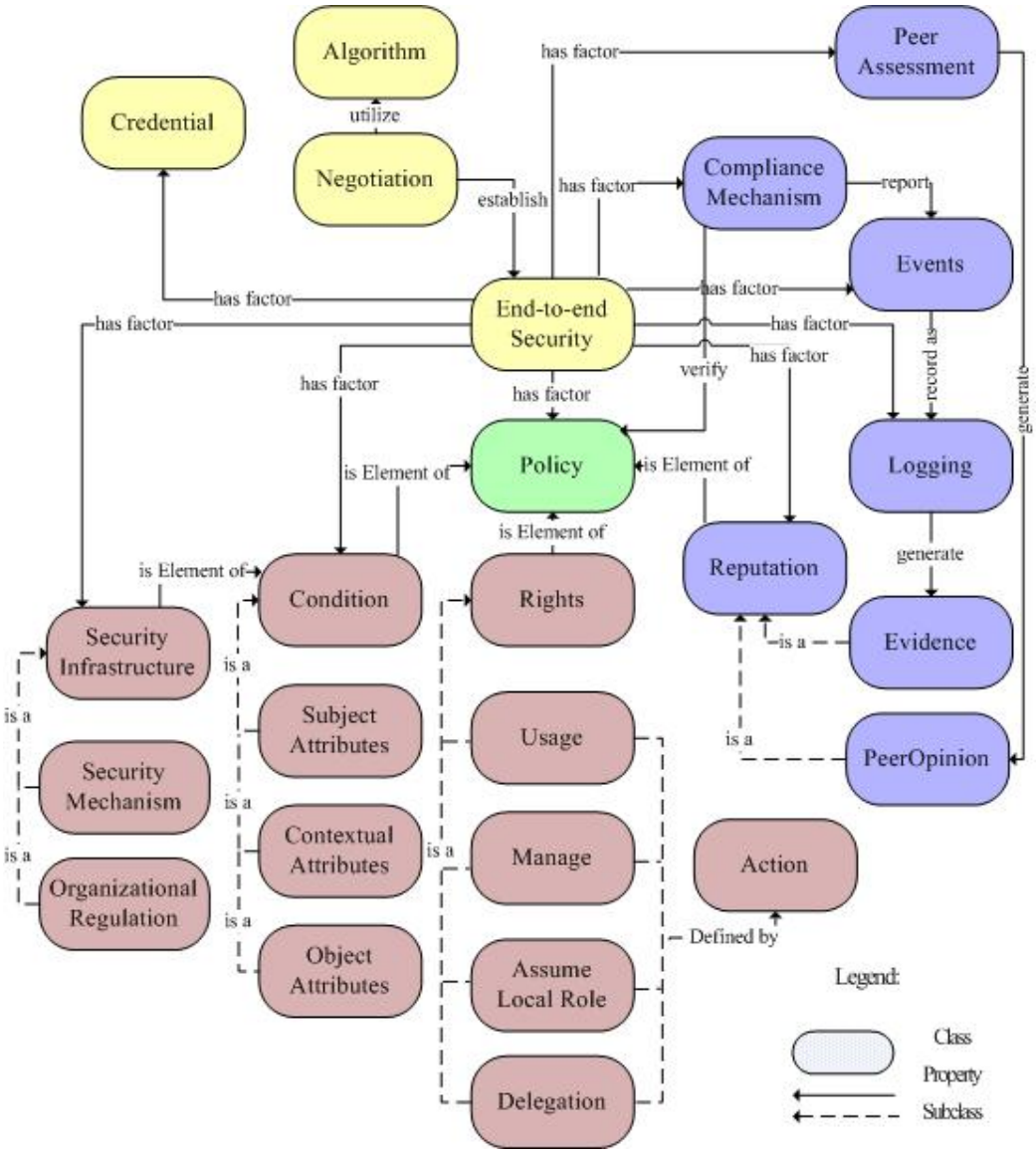


Figure 4.5: policy vocabulary

based delegation (as in RBAC) and identity based delegation (an example is the 'recipient' element in P3P).

'Condition' collects the factors that will be evaluated when deciding whether to grant usage rights. It includes inter-organizational factors:

- 'Object Attributes', which represents the attributes unique to object, for example the 'Object Type' (as 'information' or 'process');
- 'Subject Attributes', which are the attributes unique to subject;
- 'Contextual Attributes', e.g. time, location, infrastructure, etc.

- The inter-organizational factors denote the security factors in an organization, which form the security profile of the organization. Basically, there are three types of such factors:
  - 'Security Infrastructure' collects the intra-organizational factors.
  - 'Security Mechanism' denotes the traditional threats mitigation measures in information system security research.
  - 'Organizational Regulation' reflects the 'people' aspect. It denotes the 'best practices' to improve organizational security level.
- 'Compliance Mechanism' collects the possible choices of policy enforcement and monitoring mechanisms, e.g. audit log, program monitoring, agent and TPM. 'Events' in the monitoring process will be recorded as 'Logging', which serves as security-related 'Evidence'. This evidence will be composed with 'Peer Opinion' generated by 'Peer Assessment' to provide 'Reputation' records.

4.7.2 Attributes taxonomy

We define here taxonomy of security factors related to collaborative systems, in order to build the foundation of the vocabulary base. We investigate the global factors and attributes related to Information System, that is, factors that are commonly used when considering security. In this sense, our vocabulary can serve as a common base. Moreover, this basic set can be extended to integrate knowledge of specific application domains.

4.7.2.1 Subject attributes

We define four types of basic attribute names: subjectID, name, description, Role (see formula 4.46).

$$\begin{aligned} subject\_attribute\_name ::= &"subjectID"|"name"|"description" \\ &|role|extension; \end{aligned} \tag{4.46}$$

The attribute 'name' closely depends on the domain of discourse. Users can define additional subject attributes according to their specific business application domain. The attributes 'extension' represents other domain specific knowledge related to the subject. By this way, additional attributes can be defined by users. The 'description' attribute is used to record "human readable remarks" (for explanation) about the subject. The 'role' attribute is defined as:

$$\begin{aligned} role ::= &"roleof(" \\ &"entityID","=",entityID, \\ &"local\_role","=",local\_role\_name, \\ &\{role\_parameter\} \\ &")"; \end{aligned} \tag{4.47}$$

The 'entityID' denotes the organization (e.g. '*enterprise_A*') that defines a 'local role' (e.g. '*enterprise_A.manager*'). Such an approach enables the 'local role mapping' [86] [110], which is used to corelate two organizations. The 'role parameter' factor is used to express the role hierarchy as in RBAC [95]: super_role, sub_role and co_signed role. Attribute value are defined as:

$$attribute\_value ::= basic\_data\_type; \tag{4.48}$$

The 'basic_data_type' in definition (4.48) is defined as those in EPAL [15] (using 'XML Schema Part' 2 as the schema of basic data type) and XAMCL. It covers most data types in Information Systems and used on the Internet.

4.7.2.2 Object attributes

We define 7 basic object attributes, namely objectID, object name, description, object type, object category, hierarchy and object form (see formula

4.49).

$$\begin{aligned} object\_attribute\_name ::= & "objectID" | "name" | "version" \\ & | "description" | "object\_type" \\ & | "object\_category" | hierarchy \\ & | form | extension; \end{aligned} \tag{4.49}$$

We define three basic object types: '*process*', '*data*' and '*localrole*'. We model the 'local role' factor as a type of object (asset) so that the owner can express its role assignment policy. The 'category' factor is application-dependent. It is used to express 'about what kind of information' the object is. For example P3P [69] proposed 15 categories, concerning Id of entity, infrastructure (computer), past usage events, environment information, information about people and financial (payment) information. The 'hierarchy' is used to define composite object, as an 'object' can be associated to 'sub_objects' (see formula 4.50). Users can define different rules for an object and its sub_objects. The rules for sub_objects have priority to the rule for object.

$$hierarchy ::= "parent\_object" | "child\_object"; \tag{4.50}$$

The 'form' factor, as discussed in ODRL, have 5 sub factors: format, water_mark, encryption, quality, volume (see formula 4.51).

$$\begin{aligned} form ::= & \{"format"\}, \{"water\_mark"\}, \{"encryption"\}, \\ & \{"quality"\}, \{"volume"\}; \end{aligned} \tag{4.51}$$

Lastly, as for subject attributes, 'extension' is used for additional attributes defined by users to represent knowledge pertaining to specific domain of discourse, and 'description' is used for "human readable remarks" (for explanation) on the subject.

4.7.2.3 Events

Events are either 'basic event' or 'composite event' (4.52). The later is composed of the formers connected with '*relative_temporal_function*' (4.53).

$$event ::= basic\_event | composite\_event; \tag{4.52}$$

$$
\begin{aligned}
composite\_event ::= & composite\_event\_name, "(", \\
& basic\_event, \\
& relative\_temporal\_function, \\
& basic\_event, \\
& \{relative\_temporal\_function, \\
& basic\_event\} \\
& ")";
\end{aligned}
\tag{4.53}
$$

A 'Basic event' is as defined as (formulae 4.54 and 4.55):

$$
\begin{aligned}
basic\_event ::= & event\_name, [in\_session], \\
& [event\_start\_time], [event\_end\_time];
\end{aligned}
\tag{4.54}
$$

$$
event\_name ::= action\_event | session\_event | security\_event; \tag{4.55}
$$

The 'action event' (Right_exercise event) is the 'Actions' defined in 'Rights' that becomes 'Event' after it is exercised by consumer. The 'session event' and 'security event' are defined as in (4.56) and (4.57):

$$
\begin{aligned}
session\_event ::= & "request" | "negotiate" | "session\_start" \\
& | "session\_end" | "session\_evolution";
\end{aligned}
\tag{4.56}
$$

$$
\begin{aligned}
security\_event ::= & "task\_related\_security\_event" \\
& | "process\_fault" | "external\_security\_event";
\end{aligned}
\tag{4.57}
$$

The 'task_related security event' stands for the actions that cause a violation of policy. The 'process fault' is related to other faults that occur in a collaboration context, e.g. the 'standard fault' in BPEL. The 'external security event' denotes faults caused by external sources, e.g. faults in infrastructure (machine, network, etc.).

To describe the status of a dynamic collaborative system, relations between events have to be set (e.g. defining rules like "if event 'e1' and 'e2' both happen, 'e3' happens"), achieved by defining 'event relation functions':

$$
\begin{aligned}
event ::= & event, \\
& event\_relation\_function, event, \\
& \{event\_relation\_function, event\};
\end{aligned}
\tag{4.58}
$$

We group the $event\_relation\_function$ into two classes '$concurrent\_cause$' and '$sequential\_cause$':

$$event\_relation\_function ::= concurrent\_cause | sequential\_cause; \quad (4.59)$$

- **The 'concurrent_cause'** is defined as:
  One (event) cause One result (resulting event); One cause N (concurrent) results; N (concurrent) cause One result; N (concurrent) cause N (concurrent) results; One_of_ N (concurrent) cause; One_of_ N (concurrent) result.
- **The 'sequential_cause'** is defined as:
  Repeat cause (one cause repeat n times, with the same result); Temporal cause (for a event 'e0' that sustains for some time, if it sustain to time point 'T1' then it causes event 'e1', if 'e0' lasts to time point 'T2' it cause 'e2').

4.7.2.4 Context attributes

As expressed in (4.60) contextual condition consists in security infrastructure, transaction_related attributes and environment attributes.

$$\begin{aligned} context\_attribute\_name ::= & security\_infrastructure \\ & | transaction\_attributes \\ & | environmental\_attributes; \end{aligned} \quad (4.60)$$

$$\begin{aligned} security\_infrastructure ::= & \{hardware\}, \{software\}, \\ & \{deliveryChannel\}, \{people\}; \end{aligned} \quad (4.61)$$

'Security infrastructure' (4.61) covers 4 aspects: physical security (associated to the hardware), system security (related to software), network security (i.e. delivery channel) and human factors (that is, the organizational level security best practices such as those defined in EBIOS [83], OCTAVE [11], SNA [90] and ISO27002 [141]). Some works such as the NRL security ontology [163] that collects relating concepts and terms are associated to these "technical" aspects.

$$\begin{aligned} transaction\_attributes ::= & \{history\_event\}, \\ & \{current\_transaction\_event\}, \\ & \{concurrent\_transactions\_event\}; \end{aligned} \quad (4.62)$$

'Transaction related attributes' (4.62) include 'events' related to business process (see section 4.52 for definition for 'event' factor):

- 'historical event' represents the events in past terminated transaction (collaboration context).
- 'current transaction event' represents the events in '**current**' (living) transaction.
- 'concurrent transaction event' represents the events in '**other**' (living) transactions.

$$environment\_attribute ::= \{temporal\_factor\}, \{spatial\_factor\}, \{extension\}; \tag{4.63}$$

Environment attributes basically consist in 'spatial', 'temporal' and other attributes depending on the application domain, e.g. temperature, wind, humidity, electromagnetic intensity, etc.

$$spatial\_factor ::= physical\_location | logical\_location | digital\_location; \tag{4.64}$$

'Special factors' can be related to 'physical location' (e.g. a postal address; position information in longitude, latitude and altitude, etc.) and 'logical location' (which includes, for example, IP address, URI, ISBN, page number, etc).

'Temporal factor' are used in two ways in our policy model (see section 4.7.2.10 for more discussion): either used in a predicate to describe the environmental factor 'time' or associated to the predicate (or a rule) to define extra information concerning the lifecycle of the predicate (or the rule).

4.7.2.5 Rights categorization

A collaborative context policy calls for comprehensive **Right** part (see formula 4.65) defining not only access and a particular 'usage' (used to express usage/consumption actions), but also other types of rights as:

- 'management' used to express resource exchange actions,
- 'assume local role' that enables users to incorporate the effect of local RBAC policies in collaborative environment,

- 'delegation' that supports role-based delegation (as in RBAC) and identity-based delegation (an example is the 'recipient' element in P3P).

$$right ::= usage|management|assume\_local\_role|delegation; \tag{4.65}$$

$$usage ::= action; \tag{4.66}$$

$$management ::= action; \tag{4.67}$$

4.7.2.6 Action

Actions are defined as atomic action or composite action (formula 4.68). Atomic actions can be defined by a rather static 'basic action vocabulary' (see formulae 4.69 and 4.70), whereas users can define 'composite actions ' with the 'basic' actions or with other composite actions (see formula 4.71 for the construction of composite actions with atomic actions).

$$action ::= atomic\_action|composite\_action; \tag{4.68}$$

$$atomic\_action ::= atomic\_action\_name, \{parameters\}; \tag{4.69}$$

$$\begin{aligned} atomic\_action\_name ::= & "read"|"write"|"create"|"delete" \\ & |"send"|"receive"|"render"|"execute" \\ & |"encrypt"|"decrypt"|"digitally\_sign" \\ & |"verifying"|"assume\_local\_role" \\ & |"delegation\_of\_rights" \\ & |"delegation\_of\_control" \\ & |"revocation\_of\_delegation"; \end{aligned} \tag{4.70}$$

$$\begin{aligned} composite\_action ::= & composite\_action\_name, "(", \\ & atomic\_action, \\ & relative\_temporal\_function, \\ & atomic\_action, \\ & \{relative\_temporal\_function, \\ & atomic\_action\} \\ & ")"; \end{aligned} \tag{4.71}$$

A Composite action is a sequence of atomic actions "connected" by relative temporal functions. For example 'pay by credit card' can be expressed by a sequence of three atomic actions: '$read(Credit\_card, Balance)$' followed by '$execute(subtract(Credit\_card\_balance, Charge))$' which is in turn followed by '$write(Credit\_card, Balance)$'. The detail of relative temporal function is defined by formula (4.83) in section 4.7.2.10.

We define 19 normal composite actions (see 4.72), based on the survey of mainstream access control and DRM languages as: ODRL, XrML, EPAL, XACML, etc.

$$\begin{aligned} composite\_action\_name ::= & "display" | "play" | "print" | "merge" \\ & | "split" | "move" | "duplicate" | "backup" \\ & | "save" | "restore" | "install" | "uninstall" \\ & | "sell" | "give" | "lend" | "lease" \\ & | "pay" | "sign\_contract" | "register" \\ & | extension; \end{aligned} \tag{4.72}$$

The 'extension' means that a user can define additional composite actions according to different domain of discourse. The following short examples explain these 19 normal composite actions:

- Usage actions 'display/play/print' can be modeled by basic atomic action as
$$render(Content) + write(ToI/Odevice)$$
;
- Modify actions 'merge/ split' can be modeled as $read(parts\_x\_of\_content) + read(parts\_y\_of\_content) + write(to\_one\_or\_different\_file)$;
- Management action can be modeled: 'move/ Duplicate/ Backup/ Save/ Restore' as $'read + write + delete'$; 'Install/ Uninstall' as $'execute'$. For example:
$$\begin{aligned} save = & read(original\_data) + (write\_to(storage\_device.newfile) \\ & + modify(storage\_device.newfile); \end{aligned}$$
- Dissemination activities as 'sell/lend/lease' can be modeled as $delegationofright$;

- Dissemination activities as 'give' can be modeled as $delegationofcontrol$;
- E_commerce actions can be modeled as, for example:

$$\begin{aligned} Pay = & Read(account\_balance) \\ & + execute(subtract(account\_balance, charge) \\ & + Write(toaccount\_balance)). \end{aligned}$$

- 'sign_contract' can be modeled as '$create(Contract)+write(Contract)$' and 'Register' as '$create(Account) + write(Account)$'.

4.7.2.7 Delegation and revocation

The owner of the Asset can 'delegate' Rights upon the Asset to its partners. Therefore the partner can act as the owner.

$$\begin{aligned} delegation ::= & delegation\_object, delegation\_target, \\ & delegation\_type, delegation\_depth, \\ & revocation\_condition, revocation\_parameter; \end{aligned} \tag{4.73}$$

Delegation is a 'right upon right': the '$delegation\_object$' is the 'Right' and the 'Asset' it is defined on. The '$delegation\_target$' is the subject that will receive the delegation. There are two types of delegation. By granting'**Delegation of rights**', the owner of an asset can define that a subject who holds a'right' upon the asset can further delegate this right to other subjects. The delegation establishes a trust path among partners, which 'bypasses' other policies defined upon the rights, by overriding them. It can be refined by a 'Delegation depth' criterion, which defines how many subjects each delegation chain can have (in other word, how many steps a right can be passed down). '**Delegation of control**' means that the owner of an asset delegates its ownership to a subject. This kind of delegation transfers the asset ownership to the target partner of the delegation.

Revocation is the 'reverse-operation' of delegation. Used as a parameter of the 'delegation', it defines the way the delegation can be revoked. Revocation can be described by many attributes. For example authors of [301] [26] identify several parameters, namely 'dependency', 'propagation', 'resilience', 'dominance' and 'modality' and propose a detailed analysis of the 16 states caused by the combination of the parameters of revocation (see [301]). In our

model, we propose to use the 'PRESET-INDEPENDENT-GLOBAL-DENY-STRONG' revocation mode, for ensuring consistency. In our policy model, the eligibility of the subject for the 'delegation' rights is decided according to its attributes, which extends the identity-based approach in traditional access control.

4.7.2.8 Restriction

The 'Restriction' element consists in factors related to 'security infrastructure', 'environmental factor', 'bound' and factors related to policy 'enforcement'.

$$\begin{aligned} restriction ::= & security\_infrastructure | environmental\_attributes \\ & | [\text{"}bound\text{"}] | [enforcement]; \end{aligned} \tag{4.74}$$

The 'bound' (or 'cardinal') factors denote the concept of 'how many times' an event occurs. The factor 'enforcement ' defined in formula (4.75) (see section 4.7.2.9) gives the policy a 'meta-level' description ability. It can regulate not only the factors (e.g. rights and condition) related to usage control itself but also the factors related to the system that supports usage control.

4.7.2.9 Enforcement

Policy 'enforcement' is described by 3 factors, as expressed in the following formula:

$$\begin{aligned} enforcement ::= & [\text{"}enforcement\_type\text{"}, \text{" = "}, enforcement\_type], \\ & [\text{"}enforcement\_grade\text{"}, \text{" = "}, enforcement\_grade], \\ & [\text{"}enforcement\_mechanism\text{"}, \text{" = "}, \\ & \ enforcement\_mechanism]; \end{aligned} \tag{4.75}$$

'Enforcement type' defines which enforcement components will be used (see formula 4.76).

$$enforcement\_type ::= [\text{"}PEM\text{"}], [\text{"}CEM\text{"}]; \tag{4.76}$$

The 'Provider side Enforcement Module (PEM)' is responsible for message interception and access decision making. It is attached to the asset

provider. The 'Consumer side Enforcement Module (CEM)' is related to (encrypted) data containing and usage action monitoring.

PEM and CEM both receive the policy negotiation result (i.e. the 'response', see section 4.5.4) from the Aggregation Engine. They both decompose the 'Right' part of the response into a set of 'atomic actions'.

On one hand, the PEM extracts the atomic actions dealing with the 'access' action by which the asset consumer applies for the asset from provider. It compares these atomic actions with the actions the consumer submits to the provider. The access is allowed if the accessing actions match the 'atomic actions' describing asset access in the 'response'.

On the other hand, the CEM extracts the atomic actions concerning 'usage (consumption)' activities upon the assets that are 'fetched' to the consumer side system. It compares them with the actual usage actions the consumer will achieve upon the acquired asset. If there is actions that does not belong to the 'usage' actions defined in the 'response', the CEM produces a security event declaring a policy violation.

We can see that the PEM represents the functionality of 'PEP' (Policy enforcement Point) in conventional access control system, whereas the 'CEM' is like the 'rights enforcement point' of DRM technology, which monitors the consumption activities on client (consumer) side.

'Enforcement grade' describes which protection is provided to the asset regarding the 'usage' actions. In other words, it defines 'how strict' the monitoring mechanism will be (see formula 4.77).

$$\begin{aligned} enforcement\_grade ::= & [" control" | " observation"], \\ & [" inhibition" | " modification"]; \end{aligned} \tag{4.77}$$

The different elements are described as follow:

- 'control' stands for the 'secured container' (usually by encryption) provided by the CEM.
- 'observation' describes the CEM's observation of a subject's action upon the object.
- 'inhibition' illustrates that the PEM or CEM blocks actions upon an asset.
- 'modification' characterizes the options that the PEM makes for changing the status of an asset before releasing it to a subject, including 'delay' the releasing time, 'replacing' some content or 'modifying' the format (as defined in formula 4.78).

$$modification ::= [\text{"}delay\text{"}|\text{"}replace\text{"}|\text{"}modify\text{"}]; \tag{4.78}$$

'Enforcement mechanism' is defined as (see formula 4.79):

$$\begin{aligned} Enforcement\_mechanism ::= & \text{"}virtual\_mechine\text{"}, \\ & \text{"}operationg\_system\_interface\text{"}, \\ & \text{"}runtime\_system\text{"}, \text{"}application\_wrapper\text{"}, \\ & \text{"}service\_wrapper\text{"}, \text{"}message/ESB\text{"}; \end{aligned} \tag{4.79}$$

The mechanisms required by CEM are implemented according to the position they have in the software stack. These mechanisms can be ordered from lower (basic) level to higher level as:

- CPU/Virtual machine monitor, e.g. VM monitor mechanism for Grid system built with Xen hypervisor [311];
- O/S interface monitor, e.g. system call monitoring [306] [129];
- runtime system level, e.g. Java Virtual Machine monitoring [59] [137];
- application wrapper, e.g. the CA 'AppLogic' product [288] that encapsulates an application and the infrastructure it needs to run and elastically scale in a cloud.

The mechanisms required by PEM are also implemented according to the position in the software stack. These mechanisms can be ordered from lower (basic) level to higher level as:

- service wrapper, i.e. the program that wraps an application, enabling them to exposed as Web Service (PEM functionality can be implemented in the wrapper);
- message/ESB, i.e. monitoring the message exchange between service provider and consumer, e.g. the 'Petals View' component of Petals ESB [243].

4.7.2.10 Temporal factors

Temporal factors can be used in two ways(4.80):

$$\begin{aligned} temporal\_factor ::= & temporal\_attribute, temporal\_function \\ & |\text{"}lc = \text{"}, temporal\_function; \end{aligned} \tag{4.80}$$

The '*temporal_attribute*', serves as a part of a predicate set (for example confining a 'usage action' within a time scope).

The '*lc*' predicate, on the other hand, is attached to policies, rules or attribute predicates – like SAT, OAT and $CNAT$ expressions and refines them, declaring the lifecycles of the policy element it refines. For example the temporal expressions '*before June 1th*' and '*in 30 days*' mean the policy element they modified will take effect 'before an absolute natural time' and 'within a bond of natural time' separately. The value '*eot*' (end-of-transaction) of *lc*' sets the effect scope to the end of a collaborative business process. By this way, the lifecycle of a policy (or a part of it) can be extended from 'only take effect between partners that interact directly' to 'take effect after resource is granted to direct partners'. Effect of attaching '*lc*' predicate to a rule is equal to attaching it to each individual predicate in the rule.

Extending the lifecycle of a predicate, rule, or policy beyond direct partners enables a party to express and ensure its security requirements are fulfilled during the whole collaboration context. By default, if '*lc*' predicate is omitted, the policy element only takes effect between immediate partners. In such case, our policy model is similar to traditional access control policy models (which only take effect between direct partners).

There are two kinds of basic temporal functions (as defined in 4.81):

- '$absolute\_temporal\_function$' is used to get a 'time spot' ('*datatime*', '*timepoint*') or a 'duration' ('*interval*')(see 4.82), where the '*datatime*' represents 'natural time' as 'day', 'month', 'year', etc., '*interval*' is a piece of time period, '*time_point*' denotes the 'happening' of a event (see formula 4.82).
- '$relative\_temporal\_function$' is used to describe the temporal relation between two processes (or events), where 'τ' stands for the standard 'modal operators' in temporal logic (e.g. '*Next*', '*Future*', '*Until*', '*Exist*', etc), 'with' depicts that one processes start in next time step/state another process starts, 'concurrent' means that multiple process starts at the same time step/state, and 'choose' illustrates that one process is selected from many ones to start in next time step/state (see formula 4.83).

$$\begin{aligned} temporal\_function ::= & absolute\_temporal\_function \\ & | relative\_temporal\_function; \end{aligned} \tag{4.81}$$

$$absolute\_temporal\_function ::= datatime|interval|time\_point; \quad (4.82)$$

$$relative\_temporal\_function ::= \tau|with|concurrent|choose; \quad (4.83)$$

4.7.2.11 Operators

Many attributes in the 'SAT', 'OAT' and 'CNAT' attribute-family have pre-defined 'vocabularies' (see formulae (4.51) and (4.70) for example). These 'vocabularies' enumerate basic concepts that can be the possible values of these attributes. Attribute predicates are built from attribute name and these attribute values, using ***operators***. The basic operators usually used are 'equality' (=), 'comparison' (such as '<' and '>' etc.) or set operators as '∈' (for more information about the basic operators, see chapter 2). In this section, we categorize the basic operators and introduce some new operators.

There are two types of operators, namely 'function_operator' and 'predicate_operator'.

'Function_operator' is used to build a 'function' expression, which is a non-Boolean expression (in other words, the express does not deduce a Boolean value) that changes the value of the variable it operates on, e.g., '$time1 + 10Days$' (see formula 4.84).

$$\begin{aligned} function\_operator ::= & logic\_operator \\ & |pattern\_operator \\ & |role\_hierarchy\_operator \\ & |set\_operator \\ & |temporal\_operator \\ & |mathematical\_operator; \end{aligned} \quad (4.84)$$

In formula (4.84), '*role_hierarchy_operator*' is the operator to 'search' a role hierarchy (defined by a party) to get role(s) according to some ('sub-role', 'super role' or 'co-signed role') condition. Other operators are the same as the 'function operators' we summarized in chapter 2.

'Predicate_operator' is used to build a 'predicate', which is a Boolean expression for comparing attribute values (or, in other words, to defines a 'value range' upon which the consumer's attribute value is checked), e.g., '$lastAccessTime > 10Days$' (see formula 4.85).

$$\begin{aligned} predicate\_operator ::= & logical\_comparison\_operator \\ & | relation\_predicate\_operator; \end{aligned} \tag{4.85}$$

In formula (4.84), '*logical_comparison_operator*' stands for the 'predicate operators' we summarized in chapter 2. The '*relation_predicate_operator*' represents a type of operators we propose to use in our policy model.

With the development of Ontology technology, we see the possibility and the need of expressing relations among these basic concepts in a domain knowledge base. Ontology-based knowledge base built with a semantic technology allows well structured concept hierarchy. In addition to define the concept/sub-concept relation, one can specify other relations among the concepts. For example, we can define relations like '*IPSec* **employ** *encryption*'. Therefore, we propose some new operators for our policy language:

- *Contradict*: one can not have attribute 'x(e.g. a 'role')' and attribute 'y' at the same time;
- *Subconcept*: 'x' has sub-concept 'y', 'z', ...;
- *Inclusion*: 'y' has part of the properties of 'x', but 'y' is not necessarily a sub-concept of 'x';
- *combination*: 'x' has the properties of those of 'y' and 'z';
- *Opposite*: 'x' is the opposite concept of 'y';
- *Imply*: if one has attribute 'x' then it must have attribute 'y';
- *Prior_to*: 'x' is a choice prior to 'y' for some selection, e.g. security mechanism;
- *Detail_then*: 'x' is a more detailed concept then 'y' for some selection, e.g. history-events;
- *Location_distance*: the space between two locations.

These operators allow a richer semantic comparison of concepts, thus enhancing the capability of our policy language. They are rather 'general level' operators, as they are not related to specific domain of discourse. The purpose is to offer a 'basic' version of library of operators, with some guiding ideas. Thus users can further extend it with their customized language elements.

By this way users can customize their own operators depending on their needs.

4.8 Sample policies

This section gives the expression of some policies in the sample use case (see section 3.2.3) in chapter 3. Then some more comprehensive use cases are introduced.

4.8.1 Policies in 'sample use case'

Based on the policy model, we express the RoP of 'C' and QoP of 'B' in a basic usage control perspective (that is, without the 'syntax extension' for collaborative context, which will be introduced in chapter 5).

$$
\begin{aligned}
&RoP_C : \\
&\quad Rt(actionID = read \vee actionID = merge) \\
&\quad \wedge Ob(actionID = delete \wedge actionTarget = O \\
&\qquad \wedge actionTime < 30days) \\
&\quad \leftarrow \\
&\qquad OAT(objectID = E(A)) \\
&\qquad \wedge SAT(certifier = A \wedge lastAccesse < 10days)
\end{aligned}
\tag{4.86}
$$

The rule head of RoP_C contains both a 'right' part and an 'obligation' part. The rights consist in 'read' and 'merge'. The obligation is 'delete', which is further constrained with time limits. The rule body is the condition part, which contains attribute predicates in the categories of OAT and SAT. The syntax symbols 'OAT' and 'SAT' are 'syntax sugars' to ease human-reading of the policy. The attribute predicates of RoP_C are rather self-explained. This rule can be easily transformed into a XACML policy by translating correspond part into XACML syntax. The usage control policy has two parts in its head. So in policy enforcement procedure, it can be divided into two Horn-clause [192] form rules (see formula 4.87 and 4.88):

- Rule (4.87) is an authorization rule defining the condition for permitting the rights of 'read' and 'merge'.
- Rule (4.88) is an obligation rule defining that after accessing to the rights is authorized, obligation is imposed.

$$
\begin{aligned}
&Au_C: \\
&\quad Rt(actionID = read \vee actionID = merge) \\
&\quad \leftarrow \\
&\qquad OAT(objectID = E(A)) \\
&\qquad \wedge SAT(certifier = A \wedge lastAccess < 10days)
\end{aligned}
\tag{4.87}
$$

$$
\begin{aligned}
&Ob_C: \\
&\quad Ob(actionID = delete \wedge actionTarget = O \\
&\qquad \wedge actionTime < 30days) \\
&\quad \leftarrow \\
&\quad Rt(actionID = read \vee actionID = merge)
\end{aligned}
\tag{4.88}
$$

The syntax of QoP_B is similar, whereas the attribute predicates here are not 'requirements' but 'statements' on the attributes that B possesses.

$$
\begin{aligned}
&QoP_B: \\
&\quad Rt(actionID = read \vee actionID = merge) \\
&\quad \wedge Ob(actionID = delete \wedge actionTimede < 20days) \\
&\quad \leftarrow \\
&\qquad OAT(objectID = E(A)) \\
&\qquad \wedge SAT(subjectID = B \wedge certifier = A) \\
&\qquad \wedge lastAccess < 9days)
\end{aligned}
\tag{4.89}
$$

4.8.2 Digital Right Management scenario

This use case combines the requirements (and the principles to responds to them) extracted from 'sample 12','ebook scenario #1', 'ebook scenario #3', 'video scenario #1' and 'super distribution example #1' from ODRL standard [227] and 'role hierarchy' from NIST RBAC standard [95].

Use case 1: The companies 'Alice fantasy tourism' (A) and 'Cote d'Azur airline' (C) co-publish an e-book 'Guide's manu for exploring the Cote d'Azur (M)' via a publisher 'Bargain e-publisher' (B). The first 5 pages of the ebook can be viewed online for free. Only the partners of 'Cote d'Azur airline Inc.'

can purchase the e-book. The total purchase fee is 100 Euros. The revenue is split, 50% to 'Alice fantasy tourism', 30% to 'Cote d'Azur airline Inc.' and 20% to the publisher. The e-book is encrypted and must be exchanged with consumer who has a secured channel as SSL. Once purchased, its usage is limited to the 'tourist guide' and its ascendant role. Moreover, usage is restricted to one CPU, and print ability is limited to a maximum of 2 copies. Another company 'David cruise line' (D) has the right to view the e-book for free and can delegate this right to another customer.

The policies are defined as follows:

- The formula (Rule 1.1) expresses the rights definition of the first usage mode of the e-book: online free review. It expresses the requirement: 'The first 5 pages of the ebook can be viewed online for free'. Other requirements are expressed through separate rules as they deal with other usage rights of the e-book.

$$\begin{aligned} & Rt(actionID = Play \wedge Pages < 5) \\ & \leftarrow \\ & \quad OAT(ID = M) \end{aligned} \qquad \text{(Rule 1.1)}$$

- The formula (Rule 1.2) expresses the rights definition of the second usage mode of the e-book: read (after purchasing) the full version of the e-book. It expresses the requirement: 'Only the partners of 'Cote d'Azur airline Inc. can purchase the e-book' (line 5, 6). The total purchase fee is 100 Euros (line 6). The revenue is split 50% to 'Alice fantasy tourism', 30% to 'Cote d'Azur airline Inc.' and 20% to the publisher (line 4). The e-book is encrypted and must be exchanged to consumer side by secured channel as SSL (line 8). Once purchased, the use of it is limited to the 'tourist guide' (role 'gd') and its ascendant role (line 7). 'The usage (play) is bound to one CPU (line 1). Who can read ('play' to screen) the e-book can print it, limitation of the print

is 2 copies (line 2).

$$
\begin{aligned}
&line\ 1: \quad Rt(actionID = play \wedge count(CPU\_ID) = 1\\
&line\ 2: \qquad \vee actionID = print \wedge count(print) \leq 2)\\
&line\ 3: \leftarrow\\
&line\ 4: \qquad Sh(A = 50; C = 30; B = 20)\\
&line\ 5: \qquad \wedge OAT(ID = M)\\
&line\ 6: \qquad \wedge SAT(partner(S) = C \wedge pay = 100Euros\\
&line\ 7: \qquad\qquad \wedge(role(S) = gd \vee descendentRole(S) = gd))\\
&line\ 8: \qquad \wedge CNAT(deliveryChannel = "SSL")
\end{aligned}
\tag{Rule 1.2}
$$

- The formulae (Rule 1.3) and (Rule 1.4) give another way of expressing the second usage mode of the e-book. The Rule 1.2 defines the right 'read', whereas the 'print' right is defined in Rule 1.4.

$$
\begin{aligned}
&Rt(actionID = play \wedge count(CPU\_ID) = 1)\\
&\leftarrow\\
&\quad Sh(A = 50; C = 30; B = 20)\\
&\quad \wedge OAT(ID = M)\\
&\quad \wedge SAT(partner(S) = C \wedge pay = 100Euros\\
&\qquad \wedge (role(S) = gd \vee descendentRole(S) = gd))\\
&\quad \wedge CNAT(deliveryChannel = "SSL")
\end{aligned}
\tag{Rule 1.3}
$$

$$
Rt(actionID = print(O, S) \wedge count(print) \leq "2") \leftarrow play(O, S)
\tag{Rule 1.4}
$$

- The formulae (Rule 1.5) and (Rule 1.6) define the rights of 'David cruise line'. The Rule 1.5 expresses '(D) has the right to view the e-book for free'.

$$
\begin{aligned}
&Rt(actionID = play)\\
&\leftarrow\\
&\quad Sh(A = 50; C = 30; B = 20)\\
&\quad \wedge OAT(ID = M \wedge encrypted = Yes)\\
&\quad \wedge SAT(organization = D\\
&\qquad \wedge (role(S) = gd \vee descendentRole(S) = gd))\\
&\quad \wedge CNAT(deliveryChannel = "SSL")
\end{aligned}
\tag{Rule 1.5}
$$

- The Rule 1.6 expresses that '(D) has the right to delegate this read right to another customer'.

$$
\begin{aligned}
&Rt(Delegate(S, play)) \\
&\leftarrow \\
&\quad Sh(A = 50; C = 30; B = 20) \\
&\quad \wedge OAT(ID = M \wedge encrypted = Yes) \\
&\quad \wedge SAT(organization = D \\
&\qquad \wedge (role(S) = gd \vee descendentRole(S) = gd)) \\
&\quad \wedge CNAT(deliveryChannel = "SSL")
\end{aligned}
\qquad \text{(Rule 1.6)}
$$

The provider's policy consists in all the rules defined above. When a client requires the e-book, the policy enforcement component will search through the policy to find the rules that matches the request. All the rules matching the request must be applied.

4.8.3 Collaborative context scenario

This use case is similar to the 'Initial Example' in WS-BPEL standard [146] but is enhanced to incorporate more participants to integrate the requirements of security and right-protection issues that can hamper collaborative business process.

Use case 2: This use case is a collaborative business process for price inquiry. The tourism association 'Eighty days around the World' (E) inquires 'Alice fantasy tourism' (A) for the total price and arrangement of a 'Cote d'Azur and the Mediterranean package tour' for 50 persons. 'A' inquires 'Beausoleil tourist office' for fete-day information. Then 'A' produces the arrangement of 'coach tour'. It further inquires 'Cote d'Azur airline' (C) for travel arrangement (including air transport and accommodation); 'David cruise line' (D) for cruise arrangement; 'Friend-arm' (F) for assurance. 'C' provides the arrangement of 'airline' and inquires three hotels 'Generous' (G), 'Hospitable' (H) and 'Ideal'(I) for room arrangements. 'David Cruise (D)' has partnership with 'Friend-arm assurance (F)'. Other participants have no partnerships. All the participants use XML as the default format for exchanging information.

4.8.3.1 Requirements-of-Protections

The participants' Requirements-of-Protections are as follows:

- 'E' stipulates that the 'tourists' information' (abstracted as 'o') must be deleted within a week since sent out.
- 'A' wants the 'coach tour (k)' information to be exchanged with secured communication channel as 'SSL' and allows data access only to 'project manager' role of E.
- 'B' has no specific requirements to protect the 'fete day information (f)'.
- 'C' requires that the 'airline information (l)' will be deleted within 15 days since it is sent out.
- 'D' will give the 'cruise (u)' information to those who have not inquired within 1 month.
- 'F' works with a party which got a recommendation from one of F's partner. F also uses only secured communication channel 'SSL' to exchange the 'assurance (n)' information.
- 'G' requires its customers to delete the 'room information (M_G)' within 10 days since they receive it.
- 'H' requires its customers to delete the 'room information (M_H)' within 4 days since it is sent out.
- 'I' requires a secured communication channel as 'IPSec' to exchange the 'room information (M_I)'.

4.8.3.2 Quality-of-Protections

The participants' Quality-of-Protections are as follows:

- 'E' follows the 'Obligation' (e.g. deleting obtained information in some days) stipulated by asset provider. It uses secured communication channel as 'SSL'. It only allows the 'project managers', 'general manager' and 'accountant' to access the 'total price and arrangement (t)' information (the later two roles are ascendants of 'project manger' role, thus have all the 'project manager' permission). E can prove owning a 'partnership with F' delegated by 'A'.
- 'A' follows the Obligation stipulated by asset provider. It uses secured communication channel as 'SSL'. It has not inquired 'D' for price within

recent 2 months. It can prove that it has a 'partnership with F' delegated by 'D'.

- 'B' follows the Obligation stipulated by asset provider.
- 'C' keeps client information less than 7 days after a 'price inquiry' process (that means, within 7 days since the information is sent by the provider). It does not have a secured communication channel as 'SSL'.
- 'D' follows the Obligation stipulated by asset provider. It is partner of 'F'.
- 'F' deletes any client information (denote as 'x') it got within 3 days in a 'price inquiry' process. It uses secured communication channel as 'SSL'.
- 'G' follows the Obligation stipulated by asset provider.
- 'H' keeps client information (in this case the tourists' information) within 2 weeks.
- 'I' follows the Obligation stipulated by asset provider.

Their 'Requirements-of-Protections' and 'Quality-of-Protections' are expressed by policies. We present hereafter the different policies of these parties. The policies negotiation should facilitate the business federation process, which is further discussed in chapter 5.

4.8.3.3 Requirements-of-Protections

Their Requirements-of-Protections are expressed by 'RoP' policies. In RoP, the 'Sh' denotes the asset provider (the owner of the policy). The 'OAT' is the attributes of the assets. The 'SAT' is the attributes of the asset consumer. For example the tourism association E has a RoP_E which is expressed as follows (see 'RoP_E'):

- E's Requirements-of-Protection takes effect during the whole business federation context, it is applied to everyone that acquires an access to E's asset ('eot' stands for 'end of transaction').
- The asset protected with the RoP_E is 'tourist information (o)', in 'XML' format.
- E is the only owner of the asset.
- Anyone can read the asset, and merge it with other information.
- Anyone accessing 'o', or any information consisting 'o', must delete it within 7 days (As the assertion is extend with "lc=eot", it affects all C-Asset that consists in this O-Asset).

$$
\begin{aligned}
&RoP_E.(lc = eot) : \\
&\quad Rt(actionID = read \vee actionID = merge) \\
&\quad \wedge Ob(actionID = delete \wedge actionTarget = O \\
&\qquad \wedge actionTime < (7days + send(O, Sh))) \qquad (RoP_E) \\
&\quad \leftarrow \\
&\qquad Sh(E = 100) \\
&\qquad \wedge OAT(ID =' o' \wedge format = XML)
\end{aligned}
$$

Similarly, we can define other RoP as follow:

$$
\begin{aligned}
&RoP_A.(lc = eot) : \\
&\quad Rt(actionID = read \wedge actionID = merge) \\
&\quad \leftarrow \\
&\qquad Sh(A = 100) \qquad (RoP_A) \\
&\qquad \wedge OAT(ID =' k' \wedge format = XML) \\
&\qquad \wedge SAT(role(S) = E.pManager) \\
&\qquad \wedge CNAT(deliveryChannel = SSL)
\end{aligned}
$$

$$
\begin{aligned}
&RoP_B.(lc = eot) : \\
&\quad Rt(all) \\
&\quad \leftarrow \qquad (RoP_B) \\
&\qquad Sh(B = 100) \\
&\qquad \wedge OAT(ID =' f' \wedge format = XML)
\end{aligned}
$$

$$
\begin{aligned}
&RoP_C.(lc = eot) : \\
&\quad Rt(actionID = read \wedge actionID = merge) \\
&\quad \wedge Ob(actionID = delete \wedge actionTarget = O \\
&\qquad \wedge actionTime < (15days + send(O, Sh))) \qquad (RoP_C) \\
&\quad \leftarrow \\
&\qquad Sh(C = 100) \\
&\qquad \wedge OAT(ID =' l' \wedge format = XML)
\end{aligned}
$$

$RoP_D.(lc = eot)$:

$Rt(actionID = read \wedge actionID = merge)$

$\leftarrow$

$Sh(D = 100)$

$\wedge\, OAT(ID =' u' \wedge format = XML)$

$\wedge\, SAT(\neg(actionID = read \wedge actionTime < 30days))$ (RoP_D)

$RoP_F.(lc = eot)$:

$Rt(actionID = read \wedge actionID = merge)$

$\leftarrow$

$Sh(F = 100)$

$\wedge\, OAT(ID =' n' \wedge format = XML)$

$\wedge\, SAT(partner(x) = F \wedge delegate(x, partner = F, S))$

$\wedge\, CNAT(deliveryChannel = SSL)$ (RoP_F)

$RoP_G.(lc = eot)$:

$Rt(actionID = read \wedge actionID = merge)$

$\wedge\, Ob(actionID = delete \wedge actionTarget = O$

$\wedge\, actionTime < (10days + read(O, S)))$ (RoP_G)

$\leftarrow$

$Sh(G = 100)$

$\wedge\, OAT(ID =' m'_G \wedge format = XML)$

$RoP_H.(lc = eot)$:

$Rt(actionID = read \wedge actionID = merge)$

$\wedge\, Ob(actionID = delete \wedge actionTarget = O$

$\wedge\, actionTime < (4days + send(O, Sh)))$ (RoP_H)

$\leftarrow$

$Sh(H = 100)$

$\wedge\, OAT(ID =' m'_H \wedge format = XML)$

$$
\begin{aligned}
&RoP_I : \\
&\quad Rt(actionID = read \wedge actionID = merge).(lc = eot) \\
&\quad \leftarrow \\
&\qquad Sh(I = 100) \\
&\qquad \wedge OAT(ID =' m'_I \wedge format = XML) \\
&\qquad \wedge CNAT(deliveryChannel = IPSec.(lc = eot))
\end{aligned}
\tag{RoP_I}
$$

RoP_I is a bit different from the other ones, as it does not set the lifecycle of the whole assertion to 'eot' but only defines 'anyone access information consisting in I's O-Asset 'M_I' must possess IPSec'. On the contrary, the 'format=XML' predicate is not attached with a temporal factor, as 'I' only requires the direct partner to parse the O-Asset in XML format. So that it can be in any other format for other cases.

4.8.3.4 Quality-of-Protections

The partners' Quality-of-Protections are expressed by 'QoP' policies. In QoP, the 'SAT' is the attributes of the asset consumer (the QoP's owner). The OAT is the attributes of the assets the consumer requires.

$$
\begin{aligned}
&QoP_E : \\
&\quad Rt(actionID = read) \\
&\quad \wedge Ob('stipulated\_by\_AP') \\
&\quad \leftarrow \\
&\qquad OAT(ID =' t' \wedge format = XML) \\
&\qquad \wedge SAT(((role(S) = E.pManager \vee \\
&\qquad\qquad role(S) = E.gManager \vee \\
&\qquad\qquad role(S) = E.accountant) \\
&\qquad\qquad \wedge descendant(E.gManager) = E.pManager \\
&\qquad\qquad \wedge descendantRole(E.accountant) = E.pManager) \\
&\qquad\qquad \wedge partner(A) = F \wedge delegate(A, (partner = F), S) \\
&\qquad \wedge CNAT(deliveryChannel = SSL)
\end{aligned}
\tag{QoP_E}
$$

QoP_E expresses:

- E requests to 'Read' the Asset 'total price and arrangement(total)' in XML format.
- It will follow any Obligations stipulated by asset provider.
- When the asset is accessed and loaded to E's system, it allows 'gManager', 'accountant' and 'pManager' to read the asset. The later two are ascendant roles of the former.
- E has partnership with F, delegated by A.
- E has secured communication channel as 'SSL'.

$$
\begin{aligned}
&QoP_A : \\
&\quad Rt(actionID = read \wedge actionID = merge) \\
&\quad \wedge Ob('stipulated\_by\_AP') \\
&\quad \leftarrow \\
&\qquad OAT((ID =' o' \vee ID =' f' \vee \\
&\qquad\qquad ID =' v' \vee ID =' u' \vee \\
&\qquad\qquad ID =' n') \wedge format = XML) \\
&\quad \wedge SAT(\neg(actionID = read \wedge actionTime < 30days) \\
&\qquad\quad \wedge partner(D) = F \wedge delegate(D, partner = F, S)) \\
&\quad \wedge CNAT(deliveryChannel = SSL)
\end{aligned}
\qquad (QoP_A)
$$

$$
\begin{aligned}
&QoP_B : \\
&\quad Rt(actionID = read) \\
&\quad \wedge Ob('stipulated\_by\_AP') \\
&\quad \leftarrow \\
&\qquad OAT(ID =' o' \wedge format = XML)
\end{aligned}
\qquad (QoP_B)
$$

QoP_C :

$$Rt(actionID = read \wedge actionID = merge) \wedge Ob(actionID = delete \wedge actionTarget = O \wedge actionTime < (7days + send(O, Sh))) \leftarrow OAT((ID =' o' \vee ID =' m'_G \vee ID =' m'_H \vee ID =' m'_I) \wedge format = XML) \quad (QoP_C)$$

QoP_D :

$$Rt(actionID = read) \wedge Ob('stipulated\_by\_AP') \leftarrow OAT(ID =' o' \wedge format = XML) SAT(partner(D) = F); \quad (QoP_D)$$

QoP_F :

$$Rt(actionID = read \wedge actionID = merge) \wedge Ob(actionID = delete \wedge actionTarget = O \wedge actionTime < (3days + send(O, Sh))) \leftarrow OAT((ID =' x' \wedge format = XML) \wedge CNAT(deliveryChannel = SSL) \quad (QoP_F)$$

QoP_G :

$$Rt(actionID = read) \wedge Ob('stipulated\_by\_AP') \leftarrow OAT(ID =' o' \wedge format = XML) \quad (QoP_G)$$

$$
\begin{aligned}
&QoP_H : \\
&\quad Rt(actionID = read) \\
&\quad \wedge Ob(actionID = delete \wedge actionTarget = O \\
&\qquad \wedge actionTime < (14days + read(O, S))) \\
&\quad \leftarrow \\
&\quad OAT(ID =' o' \wedge format = XML)
\end{aligned}
\qquad (QoP_H)
$$

$$
\begin{aligned}
&QoP_I : \\
&\quad Rt(actionID = read) \\
&\quad \wedge Ob('stipulated\_by\_AP') \\
&\quad \leftarrow \\
&\quad OAT(ID =' o' \wedge format = XML)
\end{aligned}
\qquad (QoP_I)
$$

4.9 Conclusion

This chapter presents the basic syntax and semantics of our collaborative context oriented usage control policy model, as well as the vocabulary base that supports the policy authoring work. Several sample use cases are introduced to exhibit the basic syntax of our policy language. Our expectation for the efforts of building the policy model is two-pronged. First, the language elements are defined according to the concepts, terms, and operators from canonical DRM approaches as ODRL and XrML and according to Access control approaches as RBAC and XACML. Therefore we can claim the capability of our policy model to express traditional DRM policy and Access Control Policy.

Till now, discussions mainly concern only negotiation mechanisms between one provider and one consumer. In the next chapter, we extend the discussion to the policy aggregation in collaborative context, e.g. the scenarios especially where multiple providers co-produce an artifact for a final consumer. The analysis of the collaborative business process of this sample use case is also presented in chapter 5, with the discussion of 'Context Manager' algorithm.

Chapter 5

'Usage Control' in a collaborative context

The main characteristic of collaborative context is that partners share and merge their information assets to co-produce final artifacts, either tangible productions or intangible services. A provider wants that its rights upon its asset will be protected as the asset is reused, in its original form or merged with other asset within the collaboration context. Consequently, its usage policy must remain respected. To achieve this goal, we aggregate the RoP policies of assets when they merge, as well as the QoPs of consumers, if they access the same asset. This involves identifying which RoPs (or which QoPs) should be aggregated, according to collaboration patterns. By this way, our approach insures that the asset is only accessed by eligible parties, and consistency among the partners' policies or attributes.

This chapter introduces the policy aggregation mechanism, which allows aggregating policies and detecting potential conflicts, as well as the context management mechanism, which is used to gather partners in the context, similar to the task of 'program slicing' in program analysis.

5.1 Policy aggregation

In a federated business process scenario (e.g. collaborative enterprise, virtual enterprise, supply chain information management system use cases) partners share functionalities and information beyond their organizational boundaries to support a common business goal. As far as security is con-

cerned, partners should share a set of Service Security Level Agreement (***SSLA***), reflecting security profiles of each partner and summing up the security profile of the collaboration context. Any partner must have a security profile that matches the SSLA.

One efficacious strategy to manage the SSLA is the "upstream information provider rights consolidation". When a provider contributes to the collaboration with some assets, its RoP is aggregated to the SSLA. This policy aggregation" strategy fits the co-production working mode of the collaborative context.

Another important point is to consider the protection level, i.e. QoPs, when choosing new partners, in order to maintain 'proper' security quality for accepting more partners (especially Asset-providers).

Therefore, the SSLA consists in both the aggregated RoP and the aggregated QoP. They serves as a 'Collaboration-context Security Policy (CSP)' that represents the security profile (security policies and security attributes) of the context as a whole.

5.1.1 Asset lifecycle in business federation

Partners in a business federation have a same co-production goal. The products (final or semifinished) and production processes incorporate assets from multiple providers (see figure 5.1). These products, called 'collaboration assets' (C-Assets), inherit intellectual properties contained in the 'original assets' (O-Asset) from providers. .

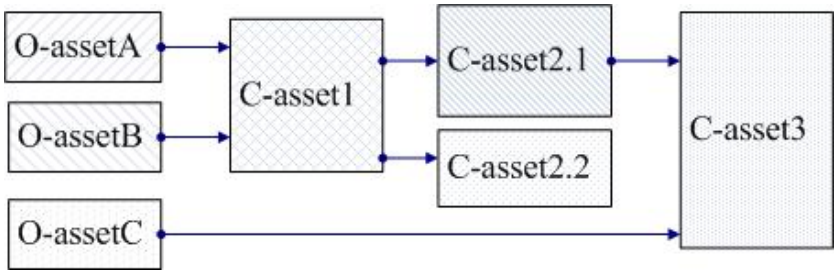


Figure 5.1: Assets derivation

Examples of inherited intellectual property traditionally protected by laws and inspection can be easily found, e.g. multimedia co-producing, joint product design, software outsourcing, technic transfer, (potentially) resource

sharing in social networks. In agile business modes supported by Information Technology, such as virtual collaborative organization Information System, new challenges appear as intellectual properties carried by digital assets, namely information (e.g. data) and processes (e.g. Web Service). They can be easily accessed and copied. Such shared information system allows high rates exchanging assets and complex patterns. As a consequence, tracking dissemination trail becomes harder. Moreover, digital assets are less tangible and unfriendly to forensic technologies. Therefore, traditional intellectual property protection may not fit this IT context. A natural thought is to explore a IT-based way out to solve this dilemma [201] [10] [267] [258]. We believe the groundwork for untangling the complexity and tracking the intractable should be the analysis of asset lifecycle and asset sharing patterns.

5.1.1.1 Asset lifecycle

In a stand-along information system context, an asset may remain isolated from others, whereas in collaborative context, an asset is often merged with others to set a new C-Asset. The O-Asset serves as a part of the C-Asset and gets a new 'derived lifecycle', separate from the 'initial lifecycle'. For example, the asset 'E: cardiac exam information' in our sample use case (see figure 3.3 in section 3.2.3) in chapter 3 is launched into the collaboration in step (3) and merged with the Asset 'M: Medical information' in step (4) (figure 5.2).

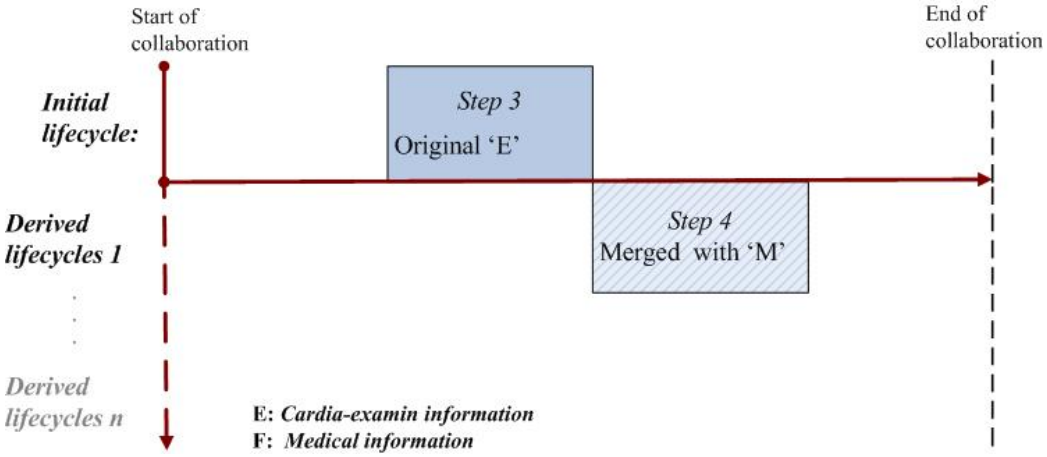


Figure 5.2: Asset lifecycle

In such a collaborative usage control scheme, the RoP policy that defines condition for using an asset (O-Asset or C-Asset) is attached to this asset. When the Asset merges with others, the RoP should merge with the other RoP, in order to protect the Asset during its whole lifecycle.

Moreover, after creating a new derived lifecycle for an asset, the former lifecycles may be terminated or may continue, according to whether the former asset will be accessed again in the future or not. In the above example, after the new lifecycle is started, the former asset is never accessed directly (but access to it must be granted as a part of the merged 'Medical information' C-Asset), so the former lifecycle is terminated.

To protect the rights of the asset provider, RoP policies should be integrated in all the lifecycles associated to the asset it protects. This is achieved by aggregating the assets' policies when they merge, so that the C-Asset inherits the policies of the assets it contains.

5.1.2 CSP management

In a collaboration process, negotiation is based on the asset consumer's QoP, denoted as QoP_{AC}, and the asset provider's RoP, denoted as QoP_{AP}. If the QoP_{AC} matches the RoP_{AP}, the corresponding collaboration context is set up, and a CSP is composed through the 'aggregation' process (see figure 5.3).

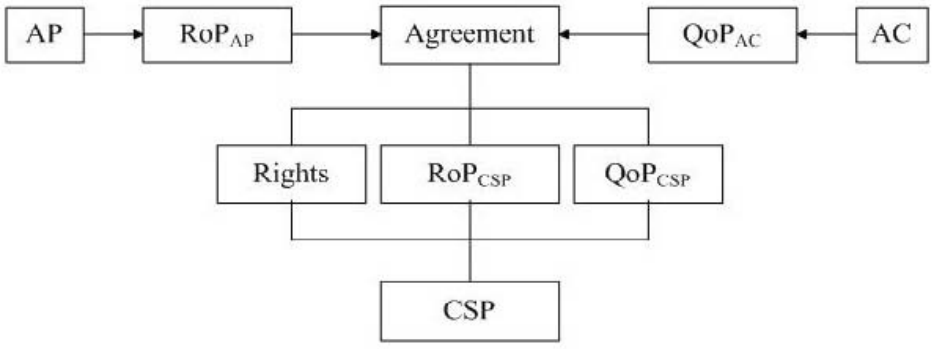


Figure 5.3: Negotiation and CSP aggregation

'Policy aggregation' is the basic idea of collaborative usage control scheme. To support the aggregation function, we extend the basic usage control scheme by adding syntax elements that allow the provider to define the policy

lifecycle and by introducing an aggregation algebra to formulate the policy aggregation mechanism. The asset sharing pattern will be discussed in the collaboration context management section (5.2).

5.1.3 Syntax extension

We use two simple indicators to extend the basic usage control policy syntax to a collaborative usage control one. The stakeholder 'Sh' and lifecycle 'lc' are introduced in the scheme definition (see definition 1 in chapter 4). On one hand, 'Sh' indicates the policy owner. It provides provenance information even after merging the original policy. On the other hand, 'lc' can be attached to a policy or to a specific attribute predicates in the policy in order to indicate the effect scope of the policy or of the predicate. While in stand-alone scheme a policy always takes effect between direct partners, in the collaborative scheme, it may take effect beyond direct partners.

For example, by tagging a policy with $lc = eot$ ('eot' represents 'end of transaction'), one stipulates that the policy should be respected during the whole business process, by anyone that consumes an artifact containing the asset associated to that policy. Those predicates that only take effect between direct partners are attached with $lc = dp$, where 'dp' stands for 'direct partner'.

We use our sample use case (see section 3.2.3) in chapter 3 to give a more featured description: RoP_C and RoP_B, considering the collaboration context, are as follows:

$$\begin{aligned} &RoP'_C.(lc = eot) : \\ &\quad Rt(actionID = read \vee actionID = merge) \\ &\quad \wedge Ob(actionID = delete \wedge actionTarget = O \\ &\qquad \wedge actionTime < 30days) \\ &\quad \leftarrow \\ &\qquad Sh(C = 100)) \\ &\qquad\quad OAT(ID = E(A)) \\ &\qquad \wedge SAT(certifier = A \wedge lastAccess < 10days) \end{aligned} \tag{5.1}$$

The basic RoP_C is tagged with '$lc = eot$'. It defines that the policy should take effect during the whole business process. '$Sh(C = 100)$' indicates that the owner of the policy is C. It is the unique owner of the object '$E(A)$' ('100' percent of ownership).

$$
\begin{aligned}
RoP'_B = \\
&Rt(actionID = read \vee actionID = merge).(lc = eot) \\
&\leftarrow \\
&\quad Sh(B = 100)) \\
&\qquad OAT(ID = M(A) \wedge encrypted = RSA).lc = eot \\
&\quad \wedge SAT(certifier = A \wedge lastAccess < 90days).(lc = eot) \\
&\quad \wedge CNAT(deliveryChannel = SSL).(lc = dp)
\end{aligned} \tag{5.2}
$$

'RoP'_B' is a bit different: '$lc = eot$' is not attached to the whole policy but to individual predicates. The '$deliveryChannel = SSL$' predicate is attached to '$lc = dp$' because the policy of B says: "(4) Those ***directly*** reading the information from B should use 'SSL' as delivery channel". When aggregating RoP_B with other policies, these predicate are processed differently: predicates tagged with '$lc = dp$' are discarded during aggregation.

There is some complexity related to policy aggregation as the foundation of aggregation is which policies are defined upon a same set of entities (or two overlapped set of entities). Then, given that the effect of a policy can be 'permit', 'deny' or 'indeterminate', the aggregation function should be able to decide if the co-effect of multiple policies is 'logically correct' (i.e. it does not lead to mis-interpretation of the goal of policy authors). To fit this requirement, we use an 'Integration Algebra' to capture the semantic foundation of policy aggregation.

5.1.4 An Integration Algebra

As multi-policy aggregation process is based on a repeated peer to peer policy integration, we first focus on the way peer to peer policy integration process is achieved by using an Integration Algebra.

Definition 2 (Integration Algebra)**.** *The Integration Algebra for collaborative policy is a tuple*

$$(P, \Sigma, \Pi_{dc}, \&) \tag{5.3}$$

where P is a set of individual policies, Σ is a set of vocabulary of attribute names and their related domains on which the policies are defined, Π_{dc} is an unary operation of domain projection, $\&$ is a binary intersection operation.

Our Integration Algebra is based on a fine-grained integration algebra introduced in [252] with a few extensions:

- The semantics of Σ and Π_{dc} are extended to multiple-policy domain.
- & has been modified to apply 'strong consensus' (see section 2.3.8 of chapter 2) for policy aggregation.

Some definitions describe basic concepts used in our policy models:

Definition 3 (Access request)**.** *Let $a_1, a_2, ...a_k$ be a set of attribute names and $dom(a_i)$ the domain of possible value of a_i, let $v_i \in dom(a_i)(1 \leq i \leq k)$. $R \equiv \{(a_1, v_1), (a_2, v_2), ..., (a_k, v_k)\}$ is a request over Σ. The set of all requests over Σ is denoted as R_Σ*

Definition 4 (3-valued access control model)**.** *A 3-valued access control policy P is a function mapping each request Q to a value in $\{Y, N, NA\}$. The Y, N and NA stands for 'Permit', 'Deny' and 'Not Applicable' decisions. Q_Y^P, Q_N^P and Q_{NA}^P denote the set of permitted, denied and not applicable requests by the P-based evaluation. $Q_{\Sigma_P} = Q_Y^P \cup Q_N^P \cup Q_{NA}^P$, $Q_Y^P \cup Q_N^P = \phi$, $Q_Y^P \cup Q_{NA}^P = \phi$, $Q_N^P \cup Q_{NA}^P = \phi$ (ensured by the 'Deny-override' combinator in our basic usage control model).*

Although access control system usually possess a 4-valued model [217] [224] (i.e. including 'Permit', 'Deny', 'Not Applicable' and 'Indeterminate', more discussion on the effects of 4-valued policy model can be found in [217] [218]), we can deem 'Indeterminate' as 'Not Applicable' at policy model level, as that our policy model uses 'Negation as failure' principle. By this way, any request that is not explicitly permitted is denied (Whereas at implementation level, 'Indeterminate' requires usually a 'certificate discovery' process to find the 'missing' attribute).

Definition 5 (Domain constraint)**.** *A domain constraint **dc** takes the form $(a_1, range_1), (a_2, range_2), ..., (a_n, range_n)$, where $a_1, a_2, ..., a_k$ are attribute names on Σ, and $range_i (1 \leq i \leq k)$ is a set of values in $dom(a_i)$. Given a request $r = (a_{r_1}, v_{r_1}), ..., (a_{r_m}, v_{r_m})$, we say that r satisfies **dc** if the following condition holds: for each $(a_{r_j}, v_{r_j}) \in r (1 \leq j \leq m)$ there exists $(a_i, range_i) \in$ **dc**, such that $a_{r_j} = a_i$ and $v_{r_j} \in range_i$.*

Domain projection **(Π_{dc})**: This operator takes the domain constraint dc as a parameter to restrict a policy to the set of requests identified by

dc (see formula 5.4). Consequently, the domain projection operator Π_{dc} delimits a set of attribute names and their corresponding value ranges. In a collaborative context such delimitation is necessary as partners from different organizations must share application domain vocabulary.

$$P_I = \Pi_{dc}(P) \Leftrightarrow \begin{cases} Q_Y^{p_I} = \{r | r \in Q_Y^P \text{ and } r \text{ satisfies } dc\} \\ Q_N^{p_I} = \{r | r \in Q_N^P \text{ and } r \text{ satisfies } dc\} \end{cases} \tag{5.4}$$

The domain constraint means that when two policies are integrated, their vocabularies must comply with each other, precisely:

$$P_I = P_1 \& P_2 \Leftarrow (\Pi_{dc}(P_1) \subseteq \Pi_{dc}(_2)) \cup (\Pi_{dc}(P_2) \subseteq \Pi_{dc}(P_1)) \tag{5.5}$$

where ***intersection*** (&) is defined as in formula (5.6):

$$P_I = P_1 \& P_2 \Leftrightarrow \begin{cases} Q_Y^{p_I} = Q_Y^{p_1} \cap Q_Y^{p_2} \\ Q_N^{p_I} = Q_N^{p_1} \cup Q_N^{p_2} \\ Q_{NA}^{p_I} = Q_Y^{p_1} \cap Q_{NA}^{p_2} \cup Q_{NA}^{p_1} \cap Q_Y^{p_2} \end{cases} \tag{5.6}$$

Given two policies P_1 and P_2, the intersection operation returns a policy P_I which is applicable to all requests having the same decisions from P_1 and P_2. That is, the integrated policy P_I resulting from the intersection of two policies P_1, P_2 targets to 'permit' (denoted as Q_Y) the requests that are 'permitted' by P_1 and P_2 at the same time. It denies (denoted as Q_N) the requests that are denied by P_1 and P_2 at the same time. For all other cases, P_I returns 'Not Applicable' (denoted as $Q_N A$).

With the 'Negation as Failure' principle, we can 'flatten' the 3-valued algebra to the domain of 'permit' and 'deny', as any 'NA' decision result in 'N' (deny) effect:

$$P_I = P_1 \& P_2 \Leftrightarrow \begin{cases} Q_Y^{p_I} = Q_Y^{p_1} \cap Q_Y^{p_2} \\ Q_N^{p_I} = Q_N^{p_1} \cup Q_N^{p_2} \cup Q_{NA}^{p_1} \cup Q_{NA}^{p_2} \end{cases} \tag{5.7}$$

The 'flattened' Integration Algebra is more explicit and reduces the complexity of aggregation algorithm design. As that a request will be permitted only when both providers permit it:

$$Q_Y^{p_1} \cap Q_Y^{p_2} \neq \phi \tag{5.8}$$

All other situations lead to the 'deny' decision.

This Integration Algebra based on the request space defines our policy integration model. Nevertheless, directly comparing the request spaces of two policies during aggregation is an intractable problem, as emulation of all requests accepted by a policy is not possible if some attributes predicates in the policy have unbounded value space (e.g. attributes defined on time, number, etc).

As a request can be seen as a vector of attribute predicates, a natural switchover is to compare directly the attribute predicates and analyze the relations between them to define the request space they covered and get answer for questions like, e.g., "Does exist any requests that are permitted by both of these policies?". This leads to study the relations between policy elements.

5.1.5 Relations between attribute predicates

For two attribute predicates of a same category (i.e. 'subject', 'object', etc.), several relations can be identified:

- **Contradict predicates**: predicates on a same attribute with disjunctive value domains. For example, two attribute predicate (upon the attribute '$lastAccess$') '$lastAccess > 10days$' and '$lastAccess < 3days$' are contradict predicates, as they defined two value ranges ('$[10,)$' and '$[0, 3]$') which have no intersection. Given that $Q_Y^{p_1} \neq \phi$ and $Q_Y^{p_2} \neq \phi$, we have $Q_Y^{p_1} \cap Q_Y^{p_2} = \phi$ only when p_1 and p_2 have contradictable attribute predicates.
- **Distinct predicates**: the attribute predicates that exist in only one rule, e.g. the '$encrypted = RSA$' in 'RoP'_C' of the sample use case (see formulae 5.1 and 5.2).
- **Common predicates**: the same predicates belonging to two rules, e.g. the '$certifier$' attribute of SAT in the sample use case (see section 3.2.3) in chapter 3.
- **Restricting predicates**: if two attribute predicates 'P_1' and 'P_2' (from different rules) have a same attribute name and if the value range of 'P_1' is a subset of the value range of 'P_2', they form a pair of '**restricting predicates**' and 'P_2' is ***restricted*** by 'P_1'. For example the '$lastAccess$' which exists in the SAT part of both RoP_B and RoP_C in the sample use case (see section 3.2.3) in chapter 3 belongs to such a case: '$lastAccess < 90days$' is restricted '$lastAccess < 10days$'.

- **Intersecting predicates**: two attribute predicates 'P_1' and 'P_2' (from different rules) having the same attribute name, with different but intersected value ranges. An example of this case is: '$lastAccess < 10days$' and '$lastAccess > 3days$'.

5.1.6 Rule similarity

According to the Integration Algebra, when aggregating two providers' policies, we focus on two problems:

- Finding out whether there exists any requests that can be permitted according to the 'permit' rules of both providers or not.
- If there are such requests, finding out if they are all 'overridden' by the deny rules of these two providers (in this case, no request can be permitted).

Therefore we define rule similarity according to a request space point of view and summarize the following types of rule relations.

- **Disjoint**: If two rules have 'contradict predicates', their request spaces can never overlap. Therefore they are related by a '**Disjoint**' relation, which means that no request can match both of them.
- **Conjoint**: If two rules have only 'common predicates', they cover a same request space. Therefore they are related by the '**Conjoint**' relation. A request matching one of them matches also the other.
- **Cover**: For two rules R_1 and R_2, 'R_1 **covers** R_2' means that all requests matching R_2 also matches R_1'. In other words, 'R_1' covers the request space of 'R_2'. This occurs when R_2' restricts the value range of attributes in R_1 or when R_2 extends R_1 with new attributes. Consequently, we can identify three cases leading to the '**Cover**' relation.
  - R_2 **restricts** R_1: each predicate in R_1 is restricted by a counterpart in R_2.
  - R_2 **refines** R_1: each predicate in R_1 has a counterpart in R_2 (with 'common predicate' relation) and in addition, R_2 has some 'distinct predicates'.
  - R_2 may both **restrict** and **refine** R_1 at the same time.
- **Overlap**: All other situations lead to '**overlap**' between the request spaces of two rules.

– When there are 'intersecting predicates' between the two rules, as none of them can cover the request space of the other defined on the attribute value domain of the 'intersection predicts').
– When both rules have 'distinct predicates'(similar to the above case, as none of them can cover the request space of the other associate to the 'distinct predicates').
– When some predicates in a rule 'R_1' restrict predicates in another rule 'R_2' and, at the same time, some predicates in 'R_2' restrict predicates in 'R_1'.
– The combination of the former 3 situations (all of the 3 situation can co-exist between two rules).

Defining rule similarity is very useful when combining multiple rules. For example, no request can match two 'permit rules' which have 'disjoint' relation, a 'deny rule' makes a 'permit rule' redundant if it 'covers' the permit rule, etc. Therefore, rule similarity relations have great impacts on our policy aggregation process.

5.1.7 Aggregation algorithm

The aggregation process results in a 'Context Security Policy' (CSP), which consists in two sub-policy sets, RoP_{CSP} and QoP_{CSP}. The RoP'_{CSP} (see formula 5.9) represents the providers' rights in the outcome of collaboration work. It's defined as a combination of RoP'_{AP} which represents the policy elements (policy, assertion or attribute predicates) that are refined with the 'lc' indicator in order to extend their lifecycle beyond the direct consumer.

The QoP_{CSP} combines the QoP_P(see 5.10) of all the participants. It represents the context's guarantee about future participants' policies.

$$RoP_{CSP} = \sum_{i=1}^{m} (RoP_{CSP} \uplus RoP'_{APi}) \tag{5.9}$$

$$QoP_{CSP} = \sum_{i=1}^{n} (QoP_{CSP} \uplus QoP_{Pi}) \tag{5.10}$$

Where

- m is the total number of asset providers in the collaboration context.

- n is the total number of partners (providers and consumers) in the collaboration context.
- '$\uplus$' is the aggregation function that uses the 'Integration Algebra' to aggregate two sets of policies.
- The RoP_{CSP} and QoP_{CSP} are empty sets at the beginning.

We discuss the aggregation function in the followings, paying attention to 'conflict' detection.

5.1.7.1 Integration of permit rules

In order to integrate two permit rules, we have to pay attention to the rights predicates. Two permit rules can only be aggregated when they have overlapping 'rights' part, i.e. there are some 'rights' which exist in both rules. If two permit rules, from different asset providers, define totally different rights, we call them '***irrelevant permit rules***'. In such cases, the aggregation fails, as that with 'negation as failure' and 'originator control' principles, a 'right' upon a C-Asset is permitted only when all the providers permit it.

When the permit rules have overlapping 'rights' part, we have to check whether they are 'disjoint' rules (that is, whether there are 'contradict predicates' between them or not, as 'contradict predicates', mean that no request can match both of rules). When two rules are 'irrelevant' or 'disjoint', there is a '**Conflict of incompatible permit rules**'.

Otherwise, (the two permit rules are '***compatible***') they can be aggregated. In this case, all 'distinct predicates' and 'common predicates' between them are added to the CSP policy. For the 'intersecting predicates' and 'restricting predicates', the new value range is computed as the intersection of the value ranges of the predicates in the two policies (see predicate 'lastAccess=10days' in formula 5.11).

With this method, we can generate the RoP of CSP in 'step 4' of the sample use case (see section 3.2.3) in chapter 3, using RoP'_B and RoP'_C:

$$
\begin{aligned}
&RoP_{CSP4}.(lc = eot) : \\
&\quad Rt(actionID = read \vee actionID == merge) \\
&\quad \wedge Ob(actionID = delete \wedge actionTarget = O \\
&\qquad \wedge actionTime = 30days) \\
&\quad \leftarrow \\
&\qquad Sh(C = 30, B = 70) \\
&\qquad\quad OAT(ID = ME(A) \wedge encrypted = RSA) \\
&\qquad \wedge SAT(certifier = A \wedge lastAccess = 10days) \\
&\qquad \wedge CNAT(deliveryChannel = SSL).(lc = dp(4))
\end{aligned}
\tag{5.11}
$$

In formula (5.11), the '$lc = dp(4)$' indicates that the predicate '$delivery\_channel = SSL$' is added in the 'aggregation step 4'. It will be discarded after next aggregation step (an 'aggregation step' is associated to one information exchange step between partners in a business process, usually generating a new 'sub-context version number', as described in section 5.2.2.4).

This procedure describes the basic aggregation process for two policies which both have only ***one*** permit rule. When aggregating two policies both having ***multiple*** permit rules, the aggregation process forms a series of permit rule pairs by selecting one rule from each policy. For each rules pair, it checks whether there is a 'conflict of incompatible permit rules' or not. If all the rules pairs have such a conflict, the aggregation fails. Otherwise, 'compatible' rule pairs are integrated to the resulting CSP policy.

5.1.7.2 Aggregating policies with deny rules

When there are deny rules in the policies that are aggregated, the 'conflict' potentiality increase. For example if all permit rules are 'covered' (or 'overridden' as defined in XACML) by deny rules, the resulting policy can not authorize any access request. Generally, there are 2 types of conflict:

- **'Positive-Negative Conflict of Modalities'**: It occurs when a subject is both authorized and prohibited for a right on an object. As we use 'Deny-override combinator' (see section (4.5.3 in chapter 4), a permit rule 'covered' by a deny rule is ***ineffective*** (its effect is 'overridden' by the deny rule). Therefore when aggregating two policies,

after computing all the 'compatible permit rule' pairs, we should check if all the 'compatible rules' are covered by some deny rules. In such a case, the resulting CSP policy won't authorize any access request and the aggregation fails.

- **'Conflict between Imperial and Authority Policies'**: It occurs when a subject is required to carry out an action by 'obligation' of a policy and is prohibited to carry out this action by another policy. If this happens during the aggregation of policies from different partners, it means that the partners have contradictable security policies. Consequently they should not be gathered in one collaboration context. When the conflict occurs between policies belonging to a single owner, it denotes the inconsistency of policy authoring. As a consequence, its correction is left to the policy owner (the 'correction' work is sometime called policy 'ratification').

5.1.7.3 Other conflicts

Some other types of conflict in a collaborative context should be handled, not by the aggregation process itself, but by other methods. We give a brief discussion of them:

- **'Conflict of Duties'**: It indicates that two actions can not be performed by the same subject, which leads to the 'Separation of Duties' principle [183]. Usually the policy author applies this principle by introducing a deny rule to her/his policy set to forbid some actions based on the past 'actions' of the subject. These 'history action' information should be deemed as attributes of the subject. In our policy model, the history information is provided according to the session events (including usage activities) collected by the system.
- **'Conflict of Interests'**: It means that a subject is not allowed to perform a same action upon two objects. This is handled with the same approach as the 'conflict of duties'. The owner of the object has just to define a deny rule to express this principle.
- **'Conflict of Priorities for Resources'**: It happens when two (or more) exclusive usage requests are made to a resource, or when the resource is associated to a limited amount, which is exceeded by the total amount demanded by all the requests. The 'pre-update' strategy of attribute management in $UCON_{ABC}$ [238] can ensure that such

conflict is avoided. This strategy allows the attributes of subject and object (and any other related entity, if needed) to be updated by the access control system immediately after the right is granted and before it is exerted.

When more than one partner can be chosen to join the collaboration context and if none of them has a policy that conflicts with the CSP, a 'general majority voting' strategy like those in [218] can be used to choose the 'most fitting' partner. This is introduced in the following section.

5.1.7.4 Aggregation recommendation

Using the 'Integration Algebra' and the 'Aggregation Algorithm' given previously, we can see that the CSP becomes more restrictive when aggregating new partners policies. Access 'conditions' for the 'C-Asset' increase, while 'rights' upon the 'C-Asset' decrease. Consequently, developing the collaboration context diminishes the chance to find eligible new participants. As a countermeasure, the partner selection functionality can be enriched to reduce this risk. If more than one partner can be chosen to join the context (that is, they all fulfill the context functional requirements), the partner with the 'more suitable' security profile is chosen, according to the 'Weighted Majority Voting' principle. It extends the 'Majority Voting' [218] method. 'Weighted' is due to the fact that, when examining properties of a policy, we give more importance to 'RoP' than to 'QoP', and inside the 'RoP' policy, the importance (among components) is weighted as: 'Condition'>'Right'>'Restriction'>'Obligation'.

Thus the procedure of examining policy properties is as follows:

- Slim RoP is used to favor the partners that makes the aggregated CSP less restrictive, thus allowing more opportunities for new partner in the future step. The identification of the 'Slim' RoP among many is based on the examination of the following policy components (where 'Slim' means fewer components or 'slimmer' components):

  (1) Slim 'Condition' elements of RoP_{CSP} means fewer attribute predicates defined with wider value bounds. With 'fewer predicates', a rule defines a 'less specific' range of entities (entities can be 'subject', 'object', 'context', 'environment', etc.), allowing the rule to match more requests (e.g. it's easier to match a rule with 5

attribute predicates than to match a rule with 50 attribute predicate). A 'wider value bounds' of an attribute predicate covers more of the value domain, thus is easier for the request to 'fall in' (match with) the bound (e.g. a value bound '$age \subset [1, 15]$' covers a larger request space than '$age \subset [1, 3]$').

(2) Rich 'Right' elements of RoP_{CSP} means having more attribute predicates or having the attribute predicates defining 'more usage opportunity' semantically, depending on the application domain.

(3) Slim 'Restriction' elements of RoP_{CSP}: is similar to the 'Condition' elements.

(4) Slim 'Obligation' elements of RoP_{CSP}: is similar to the 'Condition' elements.

- Rich QoP favors the partners that makes the aggregated CSP with slim 'right' part and rich 'condition', 'restriction' and 'obligation' parts, in order to 'fit in' more RoP policies for the future steps.

The rational of these principles is to 'slow down' the growth of the RoP_{CSP} and the reduction of QoP_{CSP}. The principles we defined steer the combination of policies in a single step. To optimize the aggregated CSP on the global scale of collaborative context (which involves multiple steps), the method presented in [188] can be used. In this system a policy is represented by a Multi-Terminal Binary Decision Diagram (MTBDD). Combining multiple policies means combining MTBBDs to form a Combined MTBDD (CMTBDD). As each CMTBDD represents an aggregation in our policy scheme, aggregation recommendation can be done by using state-of-the-art optimization methods to choose the 'best' CMTBDD with our recommendation principle ('slim RoP' and 'rich' QoP).

5.1.7.5 Conclusion

The security aspect of a collaborative context is managed through the CSP attached to this context. The RoP_{CSP} consists in the providers' RoP. This represents the provider's rights in the outcome of collaboration work. The QoP_{CSP} combines the QoP of all the participants, so that future participants' RoP_{AP} can be guaranteed by CSP.

The CSP evolves as participants join and quit. When new providers are required to join, there will be a policy negotiation between the QoP_{CSP} and the new provider's RoP_{AP}, while before a new consumer can join, the

negotiation focuses on its QoP_{AC} and the RoP_{CSP}. When a participant wants to quit, its behavior and QoP_P are examined to estimate whether it has obeyed the CSP or not. At the same time its RoP_P incorporated in CSP is used to examine other participants' behaviors upon its rights. The CSP can also change when a participant achieves a 'release' operation which abolishes its RoP_{AP} upon the asset involved in the outcome of collaborative work.

However, when new providers and consumers join the federation, the RoP_{CSP} grows quickly and the opportunity for new partner to join shrink quickly as the protection requirements get stricter. Next section presents a collaboration context management method that partitions the collaboration context into several sub-contexts and manages the 'sub-CSP' of each sub-context separately. The partition is based on the C-Assets as well as the rights upon them. In this way, a participant that accesses a right on a particular C-Asset just needs to comply with the policy of that sub-context.

5.2 Collaboration context management

When applying the aggregation method to a collaborative context, we divide the collaboration context into sub-contexts, using a method similar to 'information flow' tracking: only directly related partners (producing or consuming the same set of C-Assets) are aggregated into one sub-context. Each sub-context has its own 'sub-CSP'. Briefly, the **principles** to differentiate sub-contexts are:

1. **All participants having the same 'Rights' upon the same Asset(s) are gathered in one *sub-context***.
2. Each sub-context has its ***own sub-CSP***.
3. A participant can belong to **more than one sub-context at the same time**. It must follow the ***sub-CSP*** of all the sub-contexts it belongs to.

5.2.1 Sub-context modes

We differentiate 4 sub-context partition modes, namely 'EAOG', 'SASG', 'SAMG' and 'MAMG'(see figure 5.4).

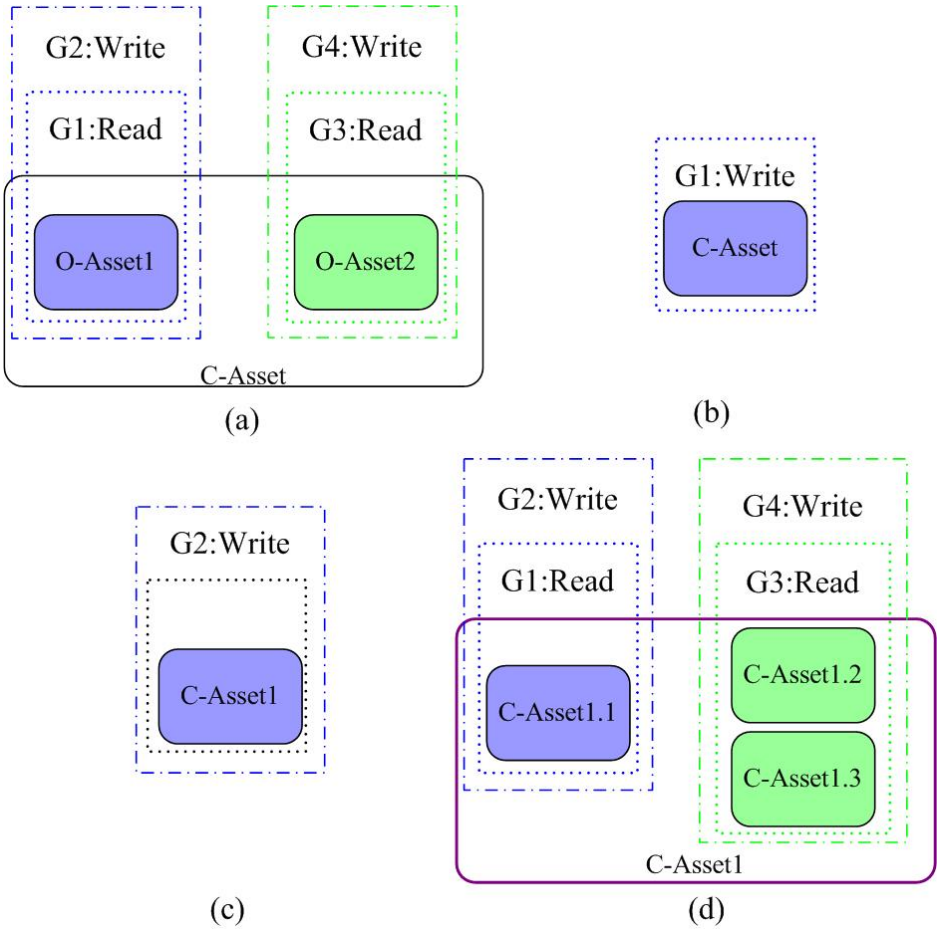


Figure 5.4: Sub-context modes. *(a)EAOG; (b)SASG; (c)SAMG; (d)MAMG.*

- **EAOG**
  When **all providers can distinguish their own O-Assets in the C-Asset** (for example when the C-Asset is a zip file which simply combines the files from some providers), they work under '***Each asset One group (EAOG)***' mode (similar to the 'stick policy' method in [208] [51] [20]). In this model each provider attaches its policy to its due part (the O-Asset) in the resulting C-Asset. Future consumers just need to obey the due policy of the part they request. In such mode there is no need for a policy aggregation algorithm. The collaboration context management is simpler. However, this mode can not fit most business federation scenarios, as usually providers' O-Assets are not simply combined together into the C-Asset but 'mixed' using some data processing procedure, making it impossible to differentiate directly each O-Asset from the C-Asset.
- **SASG**
  In the '***Single-asset Single-group (SASG)***' mode, all O-Assets are gathered in a **single group** (like in group based information sharing [191] [168] [168]). All the consumers will be given the same rights (that is, the most privileged rights required among the consumers). This is also a simple model where all policies of partners have to merge consistently in one standard 'Collaboration Context Policy' attached to the C-Asset. In such a mode, the RoP will grow quickly and conditions for new participants to join is getting harder.
- **SAMG**
  In the '***Single-asset multi-group (SAMG)***' mode, all O-Assets are gathered in a **single C-Asset**, but different consumers have different rights upon the C-asset. This is similar to group-centric information sharing [170] [169], but we add enriched rights upon information. Each right is associated to a group. In this mode, consumers holding different Rights upon the same C-Asset are assigned to different groups which have different CSPs. Participants in a sub-context share the same set of rights upon the C-Asset. By this way each sub-context's CSP are different and do not need to be compatible with others. These CSPs may evolve differently and conflicts do not need to be solved among them. If a consumer wants to access different rights upon the C-Asset, it needs to follow the CSPs of the sub-context.
- **MAMG**

The most complex mode is the'***multi-asset multi-group (MAMG)***' mode, where the O-Assets are associated to **different** C-Assets located in **different** sub-contexts. This model denotes such situation where:

- (1) there are parallel branches (caused by 'Service composition pattern 10' introduced in section 2.3.11.2 of chapter 2) in the business process and the C-Asset is split to multiple parts;
- (2) some branches are 'out-cast' branches (which do not end with 'Service composition pattern 11 and 12'–see section 2.3.11.2 in chapter 2). That is, its C-Assets will not merge into the final C-Asset.

When business process splits, the CSP is split and attached to each branch (each C-Asset) accordingly. The CSPs evolve within each branch. When processes merge, CSPs must be merged using the aggregation algorithm. If conflicts are detected during this process, the business process will halt in error and fail. To avoid this, consistency between CSPs that will merge in the future should be maintained from the splitting point. In other word, only the CSPs of the 'out-cast' branches (which do not merge with other branches) are independent.

We use a sample supply chain scenario (see figure 5.5) to illustrate the effects of these modes. This scenario has several information flows, which represent the security requirements in collaborative context: protecting upstream information providers' intellectual properties.

- Sample of the EAOG mode: when a down-stream provider (D) receives inventory information '$I_a...I_n$' from upstream providers (UP) and combines them in one XML file 'I', each of them as a separate node. The manufacturer (M) reads the nodes separately, and follows the due policy of UPs.
- Sample of the SASG mode: when production information '$C_{0a}...C_{0n}$' are blurred by 'D' to generate a global scheduling 'C_4'.
- Sample of the SAMG mode: when D blurs information '$C_{0a}...C_{0n}$' to set the scheduling (C_5) and associates it with different CSPs (e.g. differs as rights 'read' and 'store'), which are generated from UPs' policies and accessed by different down-stream partners '$D_a...D_n$'.
- Sample of the MAMG mode: if D splits a piece of information (e.g. to C_1, C_2 and C_3) and some pieces (e.g. C_3) do not merge with others in future steps.

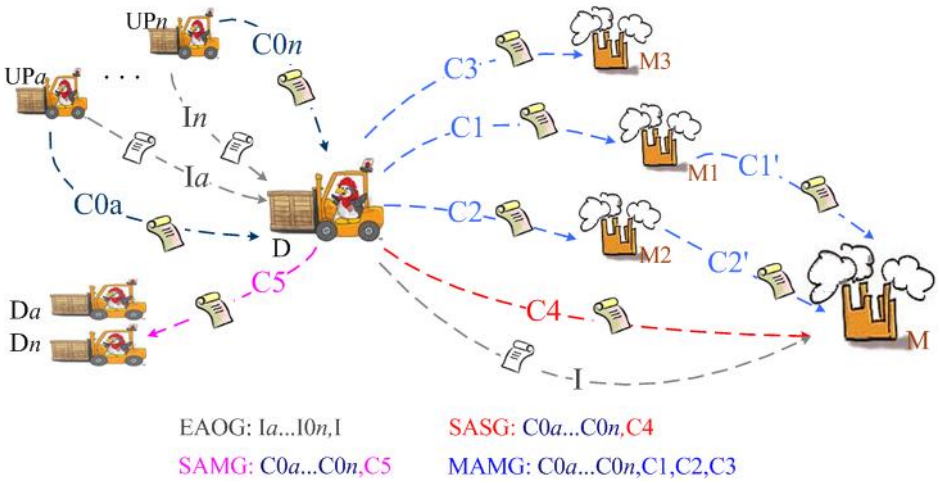


Figure 5.5: Sub-contexts patterns business federation

These 4 patterns generally exist in many collaborative contexts, as long as the issue of information asset protection and consumption exists. Imagining changing the sample case by switching the materials providers as Cloud providers or Service providers, the security management approaches still fall in our framework, but technical solution may vary.

5.2.2 Context slicing

Context slicing answers two question:

- (1) Which partners will access an asset? A question of this type for the sample use case (see section 3.2.3) in chapter 3 is "The 'Cardiac exam info' provided by C will be accessed by B or D, or both of them?"
- (2) Which assets will be accessed a party? As a simple illustration, in our sample use case (see section 3.2.3) in chapter 3, a question of this type is "Whether D will access the assets provided by B and C, either directly or indirectly".

While both of these questions can be answered intuitionally for this sample use case, the pondering procedure reflects the goal and method of context slicing. Question (1) is related to QoP aggregation among partners. Question (2) is linked to RoP aggregation. The goal is to enable the 'down-stream usage control' [47], so that indirect consumers should follow the policies of

the O-Assets involved in the C-Asset they access. The method we use is analogous to the 'Program Slicing' [117] [331] based on System Dependency Graph (SDG) [117] [122] (see section 2.3.11.1 in chapter 2).

5.2.2.1 Service call graph

The parties in a collaborative contexts are analogous to 'procedures' in a SDG. We use $P_i \xleftarrow{c} P_j$ to denote that a party P_i depends on another party P_j with 'control dependency'. $P_i \xleftarrow{d} P_j$ denotes that a party P_i depends on another party P_j with 'data dependency'. 'Data dependency' means that data provided by P_j are involved in data produced by P_i. We propose a data structure 'Service Call Graph' (SCG) based on extensions of SDG to represent partner interactions in the collaboration context.

First, 'data dependency' in a SCG is differentiated as two types: an 'aggregation dependency' means P_i involves data of P_j (the same as SDG), a 'non-aggregation dependency' denoting that data produced by P_i does not involve data from P_j (an extension of SDG). We use the sample use case (see section 3.2.3) presented in chapter 3 to illustrate these extensions (see figure 5.6):

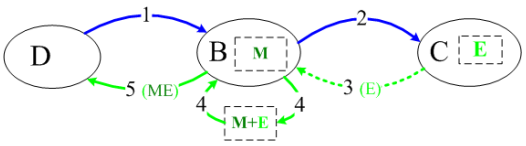


Figure 5.6: SCG of the sample use case. *In the SCG, the blue edges (step 1 and 2) represent 'control dependency'. The green edges (steps 3, 4 and 5) represent data dependency. The solid lines (edge 4 and 5) means that the output data (response) includes information from the input data (aggregation dependency). The dashed line (edge 3) means that the output data does not include information from the input data (non-aggregation dependency).*

Second, the asset carried by the message exchange is attached directly to the edges in SCG (see edges 3, 4 and 5 in figure 5.6).

Last, we represent failed interaction (due to negative result of policy negotiation) with dashed blue lines. A more comprehensive use case illustrated in section 5.2.3 will give such examples.

In order to capture assets derivation pattern, we define 'indirect dependency', based on partner service call in a business collaboration: $\forall P_i, P_j, P_k, \forall \alpha \in$

$\{c, d\}$ where P_i, P_j and P_k are partners in a collaboration, c and d are 'control dependency' and 'data dependency' relations respectively, P_i is 'indirectly dependent' on P_k, if '$P_i \xleftarrow{\alpha} P_j \wedge P_j \xleftarrow{\alpha} P_k$'.

There are two types of indirect dependency. '*Indirect* ***data*** *dependency*' is the situation where each relation in a dependency chain is 'data dependency'. We sum it up as an axiom:

Axiom 1 (Indirect data dependency)**.** $\forall P_i, P_j, P_k$*:* $P_i \xleftarrow{d} P_j \wedge P_j \xleftarrow{d} P_k \Rightarrow P_i \xleftarrow{d} P_k$

For example, in the sample use case (section 3.2.3) in chapter 3 (or, see figure 5.6 in section 5.2.2.1 or chapter 5), whether D gets the results or not depends on the response of B. B's response in turn depends on response from C.

'*Indirect* ***control*** *dependency*' is the situation where 'control dependency' relation exists in the dependency chain:

Axiom 2 (Indirect control dependency)**.** $\forall P_i, P_j, P_k, \forall \alpha \in \{c, d\}$*:* $(P_i \xleftarrow{c} P_j \wedge P_j \xleftarrow{\alpha} P_j) \vee (P_i \xleftarrow{\alpha} P_j \wedge P_j \xleftarrow{c} P_j) \Rightarrow P_i \xleftarrow{c} P_k$

As an example for indirect control dependency, in the sample use case (section 3.2.3) in chapter 3 (or, see figure 5.6 in section 5.2.2.1 or chapter 5), whether C will be called depends on B. Whether B will be called in turn depends on D. So C is indirectly control dependent on D.

We can see the slight difference between axiom 1 and axiom 2: Data dependency is transitive only when the edges in the dependency chain are all associated to 'data dependency', whereas when control dependency exists in a dependency chain, it propagates 'control dependency' to the chain.

When complex business process are defined by languages as WS-BPEL, 'Variables', which are used to carry information inside the process, must be taken into consideration, information carried by 'variables' are eventually exchanged between partners, leading to assets derivation.

These variables can be complex data type (e.g. defined by XML schema). In this case, if a part of a variable is valued-assigned to a part of another variable (see the 'sample process' in WS-BPEL specification [145]), the later variable is 'data dependent' on the former one. Thus we have the following axiom:

Axiom 3 (Direct data dependency between variables)**.** $\forall c_m \in P_i, c_n \in P_j$, $c_m \xleftarrow{d} c_n \Rightarrow P_i \xleftarrow{d} P_j$

where 'P_i' and 'P_i' stand for 'variables', 'c_m' part of 'P_i', 'c_n' part of 'P_j'.

There are only 'data dependency' relations between variables, as the only form of interactions between variable is data exchange. The conditions leading to 'indirect data dependency' between variables are the same as that for 'partners' (axiom 1).

For convenience of discussion, in the following, we don't differentiate dependency between 'partners', and 'variables', but simply call them 'partners' or 'parties'.

5.2.2.2 Service call tuple

We use a tuple $< P_i \overset{t}{\longleftrightarrow} P_j, O >$ to denote the service call from P_i to P_j, O being the set of exchanged assets. Namely we have the following types of service call tuple:

- $< P_i \overset{c}{\longrightarrow} P_j >$ denotes that P_i calls P_j with a message carrying no asset.
- $< P_i \overset{c}{\longleftarrow} P_j >$ denotes that P_i receives a message from P_j that carries no asset.
  An example of these two types of service call is when a mail agent queries a mail service for whether a mail is sent, and receives confirmation from the server. In such case the call message and the response message are deemed as not carrying any asset (i.e. information needing protection). We can see that whether a message carries asset or not depends on the straining criteria of security in a specific application context.
- $< P_i \overset{d}{\longrightarrow} P_j, O_i >$ denotes that P_i calls P_j, by sending asset O_i.
- $< P_i \overset{d}{\longleftarrow} P_j, O_o >$ denotes that P_i receives a response from P_j that carries asset O_o.
- $< P_i \overset{\alpha}{\longleftrightarrow} P_j, O_i, O_o >$ denotes that P_i calls P_j, sending asset O_i and receiving response carrying asset O_o, where O_o includes information from O_i.
- $< P_i \overset{\alpha}{\longleftrightarrow} P_j, O_i, O_o, \not\subset >$ denotes that P_i calls P_j, sending asset O_i and receiving response carrying asset O_o, where O_o does not include information from O_i.
- $< P_i \overset{f}{\longleftrightarrow} P_j, \not\subset >$ denotes that the interaction between P_i and P_j failed, due to negative result of policy negotiation.

These tuples represent the edges of SCG. Especially, the tuple $< P_i \overset{f}{\longleftrightarrow} P_j, \not\subset >$ represents the failed interaction, symbolized by dashed blue lines as mentioned before.

5.2.2.3 Assets derivation

Usually, assets derivation happens when partners interact with each other. The 'derivation' can be both 'merging' and 'splitting' assets. To build an assets derivation relation, we just need to analyze the partner interactions and generate the list of service call tuples. Basically, assets derivation occurs during partners' direct interaction. There are three situations that may incur information aggregation:

- If Y sends information to Z, who aggregates it with its own information and further send it to X. In this situation, we can identify the follow tuple sequence:

$$\begin{aligned} &< Y \overset{d}{\longrightarrow} Z, O_Y > \\ &< Z \overset{d}{\longleftrightarrow} Z, O_Y, O_Z > \\ &< Z \overset{d}{\longrightarrow} X, O_Z > \end{aligned} \tag{5.12}$$

- If Y sends information within its request to Z and gets response(s) from Z that includes Y's information. This situation is represented by the follow tuple:

$$< Y \overset{d}{\longleftrightarrow} Z, O_Y, O_Z > \tag{5.13}$$

  Extra attentions should be paid in this case, as we can not be sure that the response message includes information from the request message. Whether the output (responses) from a partner integrates the input (request) or not depends on the business logic of this partner's system. An example of this case is when Y sends some personal information to Z to calculate the insurance premium. If the response from Z consists in the insurance premium and the person's information, there is an assets derivation, otherwise (if Z answers with only the insurance premium), there is no assets derivation. In order to decide assets derivation during a direct interaction, we need information about relationships between

inputs and outputs occurring in the partner process. This can be done using partner's service functional description, e.g. WSDL in a Web Service context. It can also be done at the business process level, by adding extra indicators to a WS-BPEL script. We use the following notation to define whether the partner response includes information from request or not:

- As most of the time request information (or part of it) is included in the response, we use the default tuple to represent it:

$$< Y \overset{d}{\longleftrightarrow} Y, O_i, O_o > \tag{5.14}$$

- Whereas '$\not\subset$' is used to indicate that no information of the request is included in the response:

$$< Y \overset{d}{\longleftrightarrow} Y, O_i, O_o, \not\subset > \tag{5.15}$$

- If Y 'fetches' information from Z and aggregates its own information with it. This situation can be expressed by a service calling itself:

$$\begin{aligned} &< Y \overset{c}{\longrightarrow} Z > \\ &< Y \overset{d}{\longleftarrow} Z, O_Z > \\ &< Y \overset{d}{\longleftrightarrow} Y, O_Z, O_Y > \end{aligned} \tag{5.16}$$

As an example, we build the list of service call tuples for the sample use case (section 3.2.3) in chapter 3 (or, see figure 5.6 in section 5.2.2.1 or chapter 5) (See formula 5.17, where the tuples in the list are indexed by the steps of business process of the sample use case):

$$\begin{aligned} &< step\_1, D \overset{c}{\longrightarrow} B > \\ &< step\_2, B \overset{c}{\longrightarrow} C > \\ &< step\_3, B \overset{d}{\longleftarrow} C,' E' > \\ &< step\_4, B \overset{d}{\longleftrightarrow} B,' E',' ME' > \\ &< step\_5, D \overset{d}{\longleftarrow} B,' ME' > \end{aligned} \tag{5.17}$$

The assets derivation (merging) relation between direct partners is equivalent to 'data dependency' relation between them. The assets derivation trail (deciding sub-context pattern) is mined from the list of 'service call tuples'.

5.2.2.4 Sub-context slicing

Like the 'information reachability questions' in SDG, the assets derivation (and consumption) trail can be tracked by scanning the tuples list, in the SCG. Therefore we can ensure that providers' policies are maintained during assets derivation and are respected during asset consumption. This involves allocating assets (and providers and consumers) in sub-contexts.

We use a data structure 'context development tuple' to record the information of sub-context development: $< N_C, V_C, P_C, L_A, L_P, S_C >$, where:

- N_C is the name of the sub-context.
- V_C is its version.
- P_C the parent sub-context.
- L_A a list of all the asset involved in the sub-context.
- L_P the collection of policies in the sub context.
- S_C the step of business process.

The sub-context slicing is achieved by scanning the SCG (represented by the list of 'service call tuples') according two strategies: 'asset based slicing' and 'request based slicing'.

'**Asset based slicing**' focuses on capturing the aggregation relation among assets. Using this method, **a sub-context is created when the first O-Asset is launched** into the collaborative context by a partner. We use the RoP of this asset owner to name the context. When a new partner join the context with a new O-Asset, the sub-context consisting of the existing asset is updated, if the new partner's O-Asset is merged with the existing C-Asset. Otherwise (i.e. the new partner's O-Asset is not merged with existing C-Asset), a new sub-context is created. In our sample use case (see figure 5.6 in section 3.2.3 of chapter 3), the list of sub-context tuples is as follows:

$$\begin{aligned} &<' RoP'_C, 1, (\phi), (E), (RoP_C), step\_3 > \\ &<' RoP'_C, 2, (RoP_C.1), (E, M), (RoP_C, RoP_B), step\_4 > \\ &<' RoP'_C, 3, (RoP_C.2), (E, M), (RoP_C, RoP_B), step\_5 > \end{aligned} \tag{5.18}$$

This list describes the evolution of the sub-contexts. There is only one sub-context for the sample collaboration context, which can be represented with an assets derivation diagram (see figure 5.7).

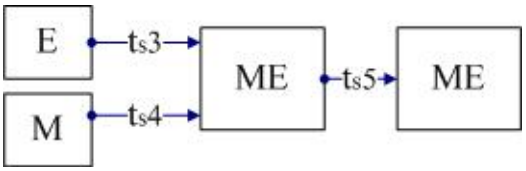


Figure 5.7: Assets derivation in the sample use case

Using the 'asset based slicing' method, we can not know whether the request from a consumer can be granted until all the assets it requests are aggregated (i.e., until the response is sent to the consumer) for answer its request. For instance, in the sample use case (section 3.2.3) in chapter 3 (or, see figure 5.6 in section 5.2.2.1 or chapter 5), D's request is initiated in '*step_*1' and whether this request will succeed or fail is decided at '*step_*4'. In case the request fails, the steps '2 and 3' are waste of partners' resources. Therefore the 'asset based slicing' method is better to be used only for 'pre-processing' a script that describes the context (e.g. WS-BPEL documents). To analyze the context on-the-fly, we can use a 'request-based slicing' method.

'**Request based slicing**', creates a sub-context **when the first request is made**. When a new partner joins the business process, either its QoP is aggregated into existing sub-context, or it leads to the creation of a new sub-context. The decision is also straight forward. **The QoPs of two partners should be aggregated, if they will access the same asset in future steps of the collaboration context**. In our sample use case (section 3.2.3) in chapter 3 (or, see figure 5.6 in section 5.2.2.1 or chapter 5), we have the list of sub-context tuples as follows:

$$\begin{aligned} &< QoP_D, 1, (\phi), (QoP_D), step\_1 > \\ &< QoP_D, 2, (QoP_D.1), (QoP_D, QoP_B), step\_2 > \\ &< QoP_D, 3, (QoP_D.2), (QoP_D, QoP_B), step\_3 > \end{aligned} \tag{5.19}$$

This tuples list captures the QoP aggregation of the context. When the first request is made by D (see '*step_*1' in formula 5.19), a sub-context is created. As B is requesting assets from C in order to respond to D, both B and D will access C's assets. Therefore, QoP_B and QoP_C are aggregated (in '*step_*2'). After '*step_*3' the list stays unchanged, as there is no new party join.

However, deciding who will access the same asset is much more tricky than it looks like, especially when partners work asynchronously, (e.g. between a partner 'X' receives request from partner 'Y' and X responds to Y, another

partner 'Z' sends request to X). We provide two basic protocols for dealing with such cases:

- Protocol 1: **After X receives request from Y, all requests that X sends to other partners P_i are deemed as *on behalf of* Y, until X responds to Y, or X receives a request from another partner Z.** This involves that a request from Y to X establishes a 'on behalf of' relation. Consequently, the QoP_Y should be aggregated into QoP_X for all the requests X sends after it receives request from Y, until X gets the result and responds to Y. However, if X accepts requests from another partner Z before X responds to Y, the 'on behalf of' relation between X and Y is interrupted by the newly established 'on behalf of' between X and Z.
- Protocol 2: **An 'X *on behalf of* Y' relation interrupted by another request from Z can be resumed after X responds to Z, if X receives a response from a partner P for a request X has sent ' *on behalf of* Y'**. This means that the 'on behalf of' relation can be nested. For example, with following request-response sequence (see the service call tuples list in formula 5.20), we can say the 'on behalf of' relation between X and Y is restored after 'X responds to Z' (step 5), because of the interaction 'P responds to X', as 'P' is a partner X has requested on behalf of Y.

$$
\begin{aligned}
&< step\_1, Y \xrightarrow{c} X, O_i > \\
&< step\_2, X \xrightarrow{c} P, O_i > \\
&< step\_3, Z \xrightarrow{c} X, O_i > \\
&< step\_4, X \xleftrightarrow{d} Q, O_i, O_o > \\
&< step\_5, X \xleftarrow{c} Z, O_o > \\
&< step\_6, P \xleftarrow{c} X, O_o > \\
&< step\_7, X \xleftarrow{c} Y, O_o >
\end{aligned}
\tag{5.20}
$$

5.2.2.5 Context development

During the context slicing process, different types of sub-context development are caused by the partners' service calls:

- **Create**: The creation of a new sub-context is always based on an independent QoP or RoP from a partner. If a partner provides an

asset which is not aggregated with other assets in the *current step*, a new sub-context consisting in the asset and the corresponding RoP is created for this *current step*. Similarly, if a partner is requesting assets on its own behalf (i.e. not because it is doing so for responding another 'former' requester) a new sub-context consisting of its QoP should be created.

- **Update**: On the contrary, 'Update' an existing sub-context happens if the partner's asset has 'data dependency' with the assets belonging to the existing sub-context, or if this partner's assets are merged with existing assets. It also happens when the partner is requesting assets on behalf of another 'former' requester, that is, it's QoP and the QoP of the former requester should be 'transmitted' to the requested party. Therefore the QoPs are in the same sub-context.
- **Merge**: 'Merge' sub-contexts is a special kind of 'update' operation. It happens when two existing assets in two sub-contexts merge, or when a partner is requesting assets on behalf of two former requesters from different sub-contexts.
- **Split**: While 'Splitting' a sub-context, several new sub-contexts are created. They all 'inherit' the assets and policies of the previous context. Context splitting can be caused by three types of interactions:
  - a party sends copies of the same asset to several partners and the copies are developed differently;
  - a party requests assets from several partners at the same time;
  - the business process has a control structure defining the execution of parallel activities.
- **End**: 'Ending' a sub-context occurs when it is 'merged', 'split' or when the business process ends.

5.2.2.6 Sub context management

As discussed previously, both context slicing methods have advantages and limits. The 'asset based slicing' fits a 'pre-processing' strategy to identify if a business process can be carried out, given the policies and attributes of partners. As far as an 'on-the-fly' context processing is used, partners' RoPs and QoPs must be aggregated as soon as they join the collaboration context. This requires using both 'asset based slicing' and 'request based slicing'.

In our sample use case (section 3.2.3) in chapter 3 (or, see figure 5.6 in

section 5.2.2.1 or chapter 5), the on-the-fly slicing first build the QoP tuples (see formula 5.19). Then from step '*step_3*', RoP tuples are listed (see formula 5.18). For managing the context on-the-fly, we manage both QoP and RoP aggregation, along with the development of the context. The decided RoP aggregation relations and QoP aggregation relations are used to generate the CSPs of each sub-context.

When a new partner joins the collaboration context, it's allocated to a sub-context according to whether it's an asset provider or consumer (or both). Then it's policies are aggregated to the CSP of that sub-context using algorithms represented by formulae 5.9 and 5.10. When sub-contexts merge, their CSPs merge, applying the same algorithms:

$$RoP_{CSP} = \sum_{i=1}^{u} \uplus(RoP_{CSPi}) \tag{5.21}$$

$$QoP_{CSP} = \sum_{j=1}^{v} \uplus(QoP_{CSPj}) \tag{5.22}$$

Next section gives a more featured illustration of the context slicing method based on the 'collaborative context scenario' (see section 4.8.3 in chapter 4).

5.2.3 Slicing a complex context

This section demonstrates the context slicing method with a more comprehensive sample use case (see the 'collaborative context scenario' introduced in section 4.8.3 of chapter 4). We introduce first business process of this use case. Then, the corresponding 'Service Call Graph' and 'service call tuple list' are described, before the 'on-the-fly' slicing process is discussed. Lastly some sample CSPs generated in the process are presented.

5.2.3.1 Collaborative business process

The business process of is (denoted as 'CBP') comprises 10 steps and two sub-process ('SBPC' and 'SBPT'), as shown in figure 5.8.

The descriptions of the steps in the CBP are as follows:

- **step 1:** 'E' sends 'tourists' information' to 'A' and query for 'total price and arrangement'.

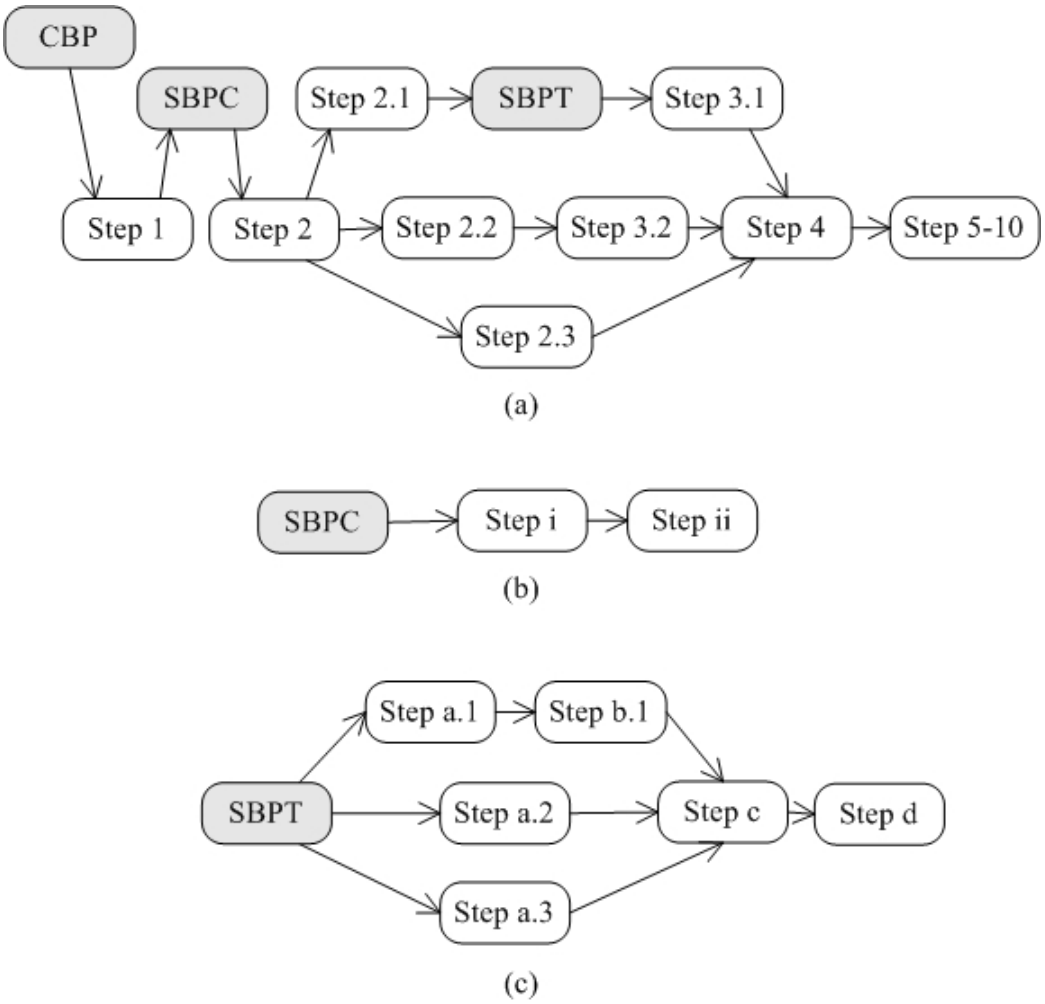


Figure 5.8: Collaborative business process example. *(a)CBP; (b)SBPC; (c)SBPT.*

- **step SBPC:** 'A' initiates a sub-process 'SBPC' to to inquiry the 'fete-day information'.
- **step 2:** It's composed of three concurrent sub-steps step ('step 2.1', 'step 2.2' and 'step 2.3'):
  - **step 2.1:** 'A' sends 'tourists' information' to 'C' to ask for 'travel' information. It is followed (indirectly, there is a sub-process 'SBPT' between them) by two sub-steps:
    - **SBPT** 'C' initiates a sub-process 'SBPT' to compose the 'travel' information.
    - **step 3.1:** 'C' sends 'travel' information to 'A'; 'C' deletes the 'tourists' information' within 7 days after CBP starts; 'C' deletes 'room' information of 'G' within 7 days after 'C' receives it.
  - **step 2.2:** 'A' sends 'tourists' information' to 'D' to query for 'cruise' information. It is followed by a step:
    - **step 3.2:** 'D' sends 'cruise' information to 'A'; 'D' deletes the 'tourists' information' within 7 days after CBP starts.
  - **step 2.3:** 'A' provides 'coach tour' information.
- **step 4:** 'A' combines 'travel', 'cruise', 'coach tour' information and 'tourists' information' to 'arrangement'.
- **step 5:** 'A' sends the combined 'arrangement' to 'F' to query for 'assurance' information.
- **step 6:** 'F' sends the 'assurance' to 'A'; 'F' deletes the 'tourists' information' within 3 days after receive (then less than 7 days after CBP starts, ensured by Aggregation Engine that managed the collaboration process); 'F' deletes 'travel' information within 3 days after 'F' receives it (so, in accordance to RoP_G that comes with '$room_G$' which is an O-Asset in the C-Asset '$travelInfo$').
- **step 7:** 'A' combines all the information to 'total price and arrangement'.
- **step 8:** 'A' sends the 'total price and arrangement' to 'E'; 'A' deletes the 'tourists' information' within 7 days after CBP starts; 'A' deletes 'travel' information within 10 days after 'A' receives it (in accordance to the combination of RoP_G and RoP_C, in this use case the RoP_G overrides the RoP_C).
- **step 9:** 'E' reads the 'total price and arrangement' within 10 days, by

the end of the 10 days, 'E' deletes it.

- **step 10:** After all participants deleted obtained asset and quit, the collaboration context terminates.

The steps in **SBPC** are:

- **step i:** 'A' sends inquiry to 'B' for 'fete-day information'.
- **step ii:** 'B' sends 'fete-day information' to 'A'.

The steps in **SBPT** are:

- **step a:** It consists in three concurrent steps initiated by 'C':
  - **step a.1:** 'C' sends 'tourists' information' to 'G' to ask for 'room' information. It is followed by:
    - **step b.1:** 'G' sends 'room' information to 'C'; 'G' deletes the 'tourists' information' within 7 days after the tourists' information is sent by 'E'.
  - **step a.2:** policy negotiation shows that the QoP_H does not satisfy RoP_E;'C' ceases calling the service of 'H'.
  - **step a.3:** policy negotiation shows that the QoP_E does not satisfy RoP_I;'I' refuses to work with 'C', because 'C' is requesting 'on behalf of' E; 'C' doesn't send 'tourists' information' to 'I'.
- **step c:** 'C' combines 'airline' information with 'room' information of 'G' and builds 'travel' information.

In the following sections, we build the Service Call Diagram (SCG) and discuss the sub-context slicing process.

5.2.3.2 Service Call Graph

The service call diagram (figure 5.9) shows the partner interaction during these steps.

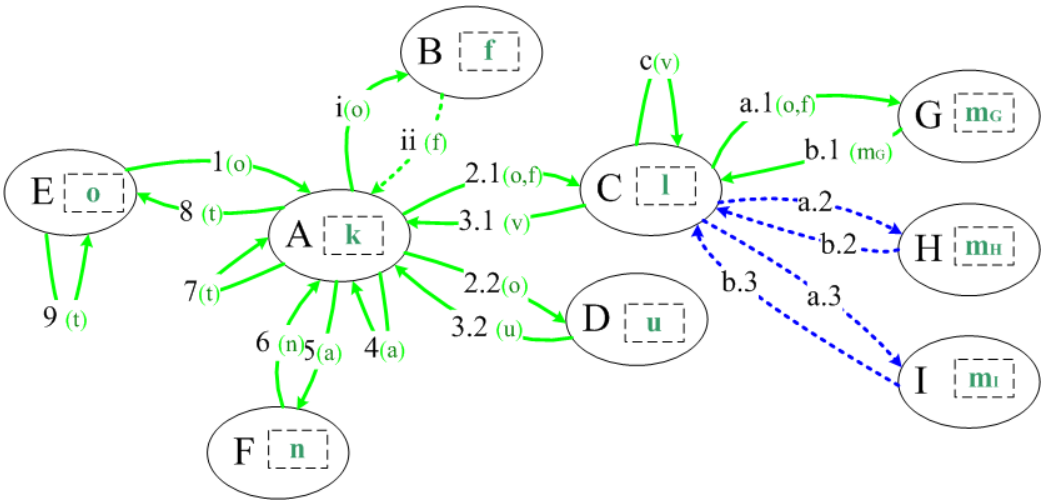


Figure 5.9: SCG of the collaborative business process. *Meanings of elements are:*

- *Green lines (1, i, ii, 2.1, a.1 b.1, c, 3.1, 2.2, 3.2, 4, 5, 6, 7, 8, 9) represent data-exchange messages which carries assets.*
- *Solid green lines (1, i, 2.1, a.1 b.1, c, 3.1, 2.2, 3.2, 4, 5, 6, 7, 8, 9) represent the output messages that have used information from the input messages.*
- *Dashed lines (ii) represent the output messages that don't include information from the input.*
- *Blue lines (a.2, b.2, a.3, b.3) represent control messages which doesn't carry assets.*
- *Dashed blue lines represent the interaction that failed, due to negative result of policy negotiation.*
- *All the green lines are attached with the asset they carry:*
  - *the data 'o' attached to step '1' stands for 'tourist information',*
  - *'f' for 'fete info',*
  - *'m' for 'room info',*
  - *'l' for 'airline info',*
  - *'v' for 'travel info',*
  - *'u' for 'cruise info',*
  - *'k' for 'coach info',*
  - *'p' for 'price',*
  - *'n' for 'assurance',*
  - *'t' for 'total price and arrangement'.*
- *Each partner also indicates the asset it provides.*

5.2.3.3 Service call tuple list

The list of service call tuple is as follows:

$$
\begin{aligned}
&<step\_1, E \xrightarrow{d} A, (o)> \\
&<step\_i + ii, A \xleftrightarrow{d} B, (o), (f), \not<> \\
&<step\_2.1, A \xrightarrow{d} C, (o)> \\
&<step\_a.1 + b.1, C \xleftrightarrow{d} G, (o), (m_G)> \\
&<step\_a.2 + b.2, C \xleftrightarrow{f} H, \not<> \\
&<step\_a.3 + b.3, C \xleftrightarrow{f} I, \not<> \\
&<step\_c, C \xleftrightarrow{d} C, (m_G, l), (v)> \\
&<step\_3.1, A \xleftarrow{d} C, (v)> \\
&<step\_2.2 + 3.2, A \xleftrightarrow{d} D, (o), (u)> \\
&<step\_4, A \xleftrightarrow{d} A, (v, u, k), (a)> \\
&<step\_5 + 6, A \xleftrightarrow{d} F, (a), (n)> \\
&<step\_7, A \xleftrightarrow{d} A, (a, n), (t)> \\
&<step\_8, E \xleftarrow{d} A, (t)> \\
&<step\_9, E \xleftrightarrow{d} E, (t)>
\end{aligned}
\tag{5.23}
$$

5.2.3.4 On-the-fly slicing

The business process starts with E's request. A QoP aggregation starts from the QoP_E. The request carries the 'tourist information' with it. Therefore an assets derivation (i.e. RoP aggregation) sub-context starts from RoP_E.

Firstly, we track the QoP aggregation process. By step 1, QoP_A is aggregated with QoP_E, to set a sub-context:

$$< QoP_E, 1, (\phi), (QoP_E, QoP_A), step\_1 > \tag{5.24}$$

The tuple denotes that the sub-context is created at *step_1* and named QoP_E, version '1'. This sub-context has no parent context (denoted by'ϕ') and includes QoPs of both E and A.

This sub-context splits as 'A' calls 'B', 'C', 'D' and 'F' separately. Thus we have 4 parallel sub-contexts (see figure 5.10).

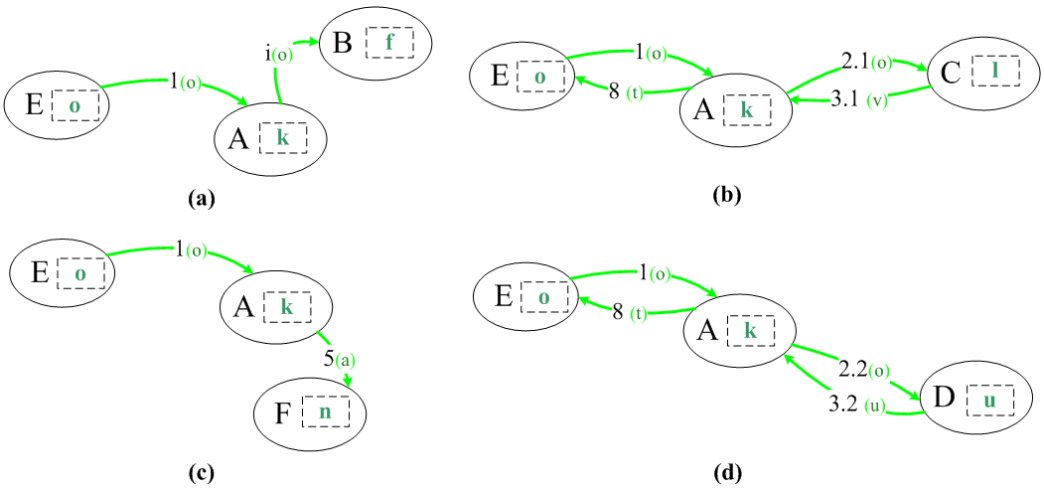


Figure 5.10: QoP aggregation. *(a) E, A, B (b) E, A, C (c) E, A, F (d) E, A, D*

They are presented by the following sub-context development tuples:

$$\begin{aligned}
&< QoP_B, 1, (QoP_E.1), (QoP_E, QoP_A), step_i > \\
&< QoP_C, 1, (QoP_E.1), (QoP_E, QoP_A, QoP_C), step_2.1 > \\
&< QoP_D, 1, (QoP_E.1), (QoP_E, QoP_A), step_2.2 > \\
&< QoP_F, 1, (QoP_E.1), (QoP_E, QoP_A, QoP_F), step_5 >
\end{aligned} \tag{5.25}$$

As 'B' and 'D' only act as 'Asset provider' in this context (they do not consume any asset from others), their QoP don't need to be aggregated.

The sub-context $QoP_C.1$ further splits into 3 three new sub-contexts, as 'C' calls 'G', 'H' and 'I' for information (see figure 5.11).

$$\begin{aligned}
&< QoP_G, 1, (QoP_C.1), (QoP_E, QoP_A, QoP_C), step_i > \\
&< QoP_H, 1, (QoP_C.1), (QoP_E, QoP_A, QoP_C), step_2.1 > \\
&< QoP_I, 1, (QoP_C.1), (QoP_E, QoP_A, QoP_C), step_2.2 >
\end{aligned} \tag{5.26}$$

This list of sub-context development tuples help us to get the decision, in 'step a.3' of 'SBPT', that 'I' can't work in this context, because QoP_E

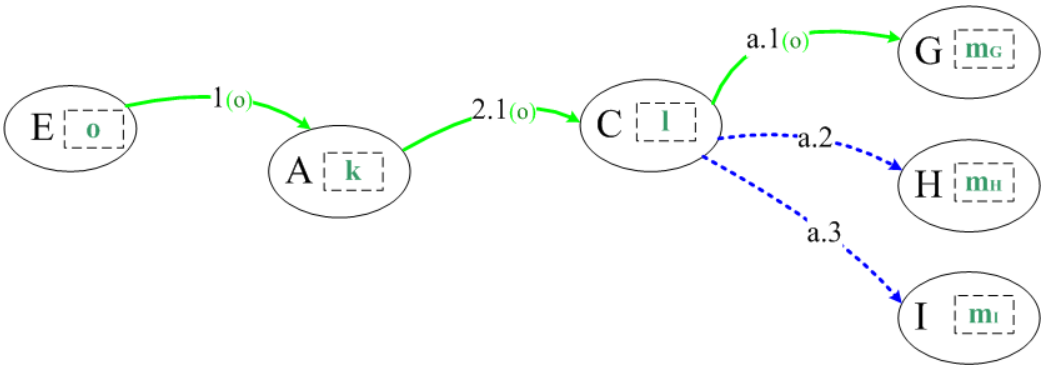


Figure 5.11: Details of QoP aggregation along 'E, A, C'

doesn't satisfy I's RoP. Otherwise, if we only use RoP aggregation, we will find out that QoP_E doesn't meet RoP_I only at 'step 8'.

In short, the QoP aggregation 'transmits' the up-stream requesters' QoPs along the business process, so policies which do not match can be found in time. However, QoP aggregation isn't sufficient. For example, with the tuple $< QoP_F, 1, (QoP_E.1), (QoP_E, QoP_A, QoP_F), step\_5 >$ we can't tell whether 'F' can join the business process or not, as the different RoPs that 'F' should meet are not provided. We use RoP aggregation to handle this.

The RoP aggregation starts with the request from 'E', as the request carries the asset 'o'. Therefore an assets derivation sub-context starts from RoP_E.

$$< RoP_E, 1, (\phi), (o), (RoP_E), step\_1 > \tag{5.27}$$

By 'step ii', as B's response 'f' doesn't contain 'o', a new sub-context is created.

$$< RoP_B, 1, (RoP_E.1), (f), (RoP_B), step\_ii > \tag{5.28}$$

However, these two sub-contexts should merge, as 'A' aggregates 'f' and 'o' before sending them to 'C' and 'D'.

$$< RoP_E, 2, (RoP_E.1, RoP_B.1), (o, f), (RoP_E, RoP_B), step\_ii > \tag{5.29}$$

This sub-contexts are succeeded by two different sub-contexts, given in

formulae (5.30) and (5.31), as 'A' calls 'C' and 'D' in parallel.

$$\begin{aligned} &< RoP_G, 1, (RoP_E.2), (m_G), (RoP_E, RoP_B, RoP_G), step\_b.1 > \\ &< RoP_G, 2, (RoP_G.1), (m_G, l, v), (RoP_E, RoP_B, RoP_G, RoP_C), step\_c > \end{aligned} \tag{5.30}$$

$$< RoP_D, 1, (RoP_E.2), (u), (RoP_E, RoP_B, RoP_D), step\_3.2 > \tag{5.31}$$

It can be seen from formulae (5.30) that the new sub context is created after 'G' answers with 'm_G'. These two sub-contexts are merged when 'A' merges all the information together.

$$\begin{aligned} &<RoP_A, 1, (RoP_G.2, RoP_D.1), (u, v, k, a), \\ &\quad (RoP_E, RoP_B, RoP_G, RoP_C, RoP_D, RoP_A), step\_4 > \end{aligned} \tag{5.32}$$

Now 'A' calls 'F' with asset 'a' and the aggregated RoP of 'E', 'B', 'G', 'C', 'D' and 'A'. As QoP_F meets with the aggregated RoP, 'F' will join the context, providing asset 'n'. This updates the sub-context '$RoP_A.1$'

$$\begin{aligned} &<RoP_A, 2, (RoP_A.1), (n), \\ &\quad (RoP_E, RoP_B, RoP_G, RoP_C, RoP_D, RoP_A, RoP_F), step\_6 > \end{aligned} \tag{5.33}$$

Then 'A' merges 'n' with 'a'.

$$\begin{aligned} &<RoP_A, 3, (RoP_A.2), (n, a, t), \\ &\quad (RoP_E, RoP_B, RoP_G, RoP_C, RoP_D, RoP_A, RoP_F), step\_7 > \end{aligned} \tag{5.34}$$

After 'step 7' the context remains unchanged, as no new provider nor consumer is joining the process. The sub-contexts capturing the RoP development is analogous to figure 5.8.

The on-the-fly strategy tracks the business process guided by QoP and RoP aggregation. QoP aggregation allows transmitting former requesters' security attributes down-stream. It discovers un-matched RoPs in time. RoP aggregation propagates former providers' security requirements down-stream and ensures no leakage of the protected information to unauthorized consumer.

5.2.3.5 CSPs

As representative examples, we give the CSPs of 'step i', 'step 5' and 'step 6'.

In '**step i**' of sub process 'SBPC', the RoP_{CSP} comes from RoP_B:

$$\begin{aligned} & RoP_i.(lc = eot) : \\ & \quad Rt(all) \\ & \quad \leftarrow \\ & \quad Sh(B, 100) \\ & \quad \wedge OAT(ID =' f' \wedge format = XML) \end{aligned} \tag{5.35}$$

The QoP_{CSP} comes from QoP_A and QoP_E:

$$\begin{aligned} & QoP_A : \\ & \quad Rt(actionID = read \wedge actionID = merge) \\ & \quad \wedge Ob('stipulated\_by\_AP') \\ & \quad \leftarrow \\ & \quad OAT((ID =' f' \wedge format = XML) \\ & \quad \wedge SAT(\neg(actionID = read \wedge actionTime < 30days) \\ & \quad \quad \wedge partner(D) = F \wedge delegate(D, partner = F, S)) \\ & \quad \wedge CNAT(deliveryChannel = SSL) \end{aligned} \tag{5.36}$$

$$\begin{aligned} & QoP_E : \\ & \quad Rt(actionID = read) \\ & \quad \wedge Ob('stipulated\_by\_AP') \\ & \quad \leftarrow \\ & \quad OAT(ID =' t' \wedge format = XML) \\ & \quad \wedge SAT(((role(S) = E.pManager \vee \\ & \quad \quad role(S) = E.gManager \vee \\ & \quad \quad role(S) = E.accountant) \\ & \quad \quad \wedge descendantRole(E.gManager) = E.pManager \\ & \quad \quad \wedge descendantRole(E.accountant) = E.pManager) \\ & \quad \quad \wedge partner(A) = F \wedge delegate(A, partner = F, S)) \\ & \quad \wedge CNAT(deliveryChannel = SSL) \end{aligned} \tag{5.37}$$

In 'step 5', the RoP_CSP is the logical combination of 'RoP_A', 'RoP_B', 'RoP_C', 'RoP_D' and 'RoP_G'.

$$
\begin{aligned}
&RoP_6.(lc = eot):\\
&\quad Rt(actionID = read \wedge actionID = merge)\\
&\quad \wedge Ob(actionID = delete \wedge actionTarget = O\\
&\qquad \wedge actionTime < (15days + send(O, Sh))\\
&\qquad \wedge actionTime < (10days + read(O, S)))\\
&\quad \leftarrow \qquad (RoP_9)\\
&\quad Sh(A, C, D, F, G)\\
&\quad \wedge OAT(ID =' a' \wedge format = XML)\\
&\quad \wedge SAT(\neg(actionID = read \wedge actionTime < 30days)\\
&\qquad \wedge role(S) = E.pManager)\\
&\quad \wedge CNAT(deliveryChannel = SSL)
\end{aligned}
$$

The QoP_CSP comes from the 'QoP_E', 'QoP_A'and 'QoP_F'.

In 'step 6', the RoP_CSP is the logical combination of 'RoP_A', 'RoP_B', 'RoP_C', 'RoP_D', 'RoP_F' and 'RoP_G'.

$$
\begin{aligned}
&RoP_6.(lc = eot):\\
&\quad Rt(actionID = read \wedge actionID = merge)\\
&\quad \wedge Ob(actionID = delete \wedge actionTarget = O\\
&\qquad \wedge actionTime < (15days + send(O, Sh))\\
&\qquad \wedge actionTime < (10days + read(O, S)))\\
&\quad \leftarrow\\
&\quad Sh(A, C, D, F, G) \qquad (RoP_9)\\
&\quad \wedge OAT(ID =' t' \wedge format = XML)\\
&\quad \wedge SAT(\neg(actionID = read \wedge actionTime < 30days)\\
&\qquad \wedge partner(x) = F \wedge delegate(x, partner = F, S)\\
&\qquad \wedge role(S) = E.pManager)\\
&\quad \wedge CNAT(deliveryChannel = SSL)
\end{aligned}
$$

The QoP_CSP comes from the 'QoP_E' and 'QoP_A'.

5.3 Conclusion

Policy aggregation is a new topic of research. It is a natural extension of 'policy interaction analysis'. 'Policy aggregation' and 'policy interaction analysis' use similar methods for detecting conflicts. Our policy aggregation is based on the investigation of some recent works of policy interaction analysis.

For tracking asset evolvements, the 'Context slicing' mechanism is developed according to a same principle as the one used in 'down-stream information flow' control. We trace the assets derivation (during information flows merging and splitting) pattern in order to ensure the consistency between the policies of providers (or consumers) for the same artifact.

By now we have introduced the major elements of our Collaborative Usage Control scheme, including the abstract syntax, semantics, vocabulary, policy aggregation mechanism and collaborative context management method. In the following chapter we introduce the functional components of our system architecture which supports the implementation of such scheme.

Chapter 6

Implementation Architecture

The enforcement of the policy model resides in implementing a DRM management layer upon an ESB (see figure 6.1). Our implementation is based on the open source projects PEtALS ESB[243] and PEtALS Master [244]. The former is a distributed standards-compliant open source ESB which includes also policy deployment and monitoring functionalities. The later is built as an upper layer and offers components for service registration, information repository and SLA (contract) management.

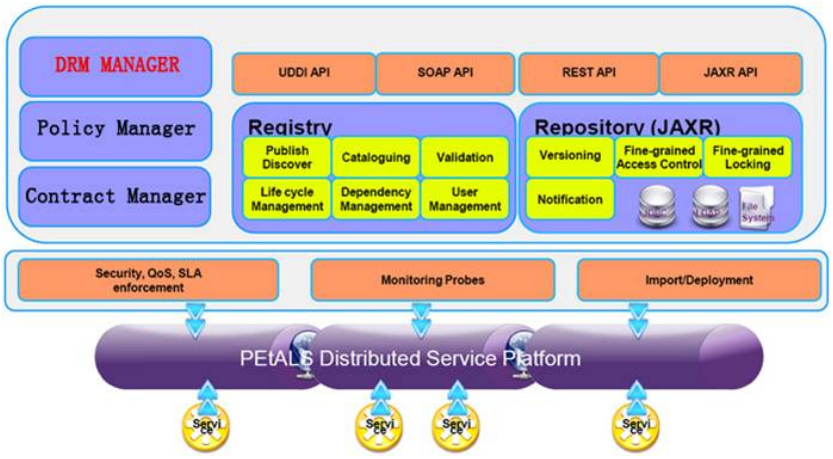


Figure 6.1: Implementation

Our prototype consists in composing end-to-end security factors to add a 'DRM Manager' layer over Petals Master (included in the Petals Security

part) to support the usage control policy model over the federation management component. This new layer implements the conceptual components described in chapter 3, as Web Services using the PEtALS ESB and PEtALS Master toolkits. These toolkits provide necessary components for message routing between services, service communication monitoring, service management, logging, etc. Leveraging these toolkits allows us to focus on the core functionalities of the new components. In the followings we introduce the overall design of the implementation architecture to provide a holistic view of the components needed in collaborative usage control.

6.1 Architecture overview

The architecture of the collaborative usage control system is shown in figure (6.2). The key components are '**context manager**', '**PDP** (**policy decision point**)', '**PGP** (**policy aggregation point**)', '**monitor service**' and '**water marking** service'. Only information flow (both data and control flows) related to security management are depicted in this figure: business process control relation (e.g. the relation between the orchestration engine and provider/consumer) are omitted.

The **context manager** is the driving component of the security management. It analyzes the status of the collaboration context for the 'sub context slicing' task. This can be done by parsing the business process script (e.g. WSBPEL file) or by on-the-fly communication with the orchestration engine to acquire the context status. It analyzes the provider/consumer pair and the exchanged Asset, tracking the asset aggregation (and consumption) patterns. Based on these data, it decides which partners' policy should be negotiated or aggregated.

The '**PDP**' is a web service which implements an XACML PDP, as our policy model is implemented with XACML. It evaluates requests (generated from QoPs) with the RoP policies.

The '**PGP**', can be used to aggregate RoPs (or QoPs) together, given no conflict is detected between the RoPs (or QoPs).

The **monitor service** has two components. An 'intrusive module' resides in the consumer system for monitoring consumer side application activities when consuming data. A component external to the consumer system (e.g. at ESB level) 'intercepts' the context level message exchange, as these messages reveal information about the partners. Detail discussion about these two

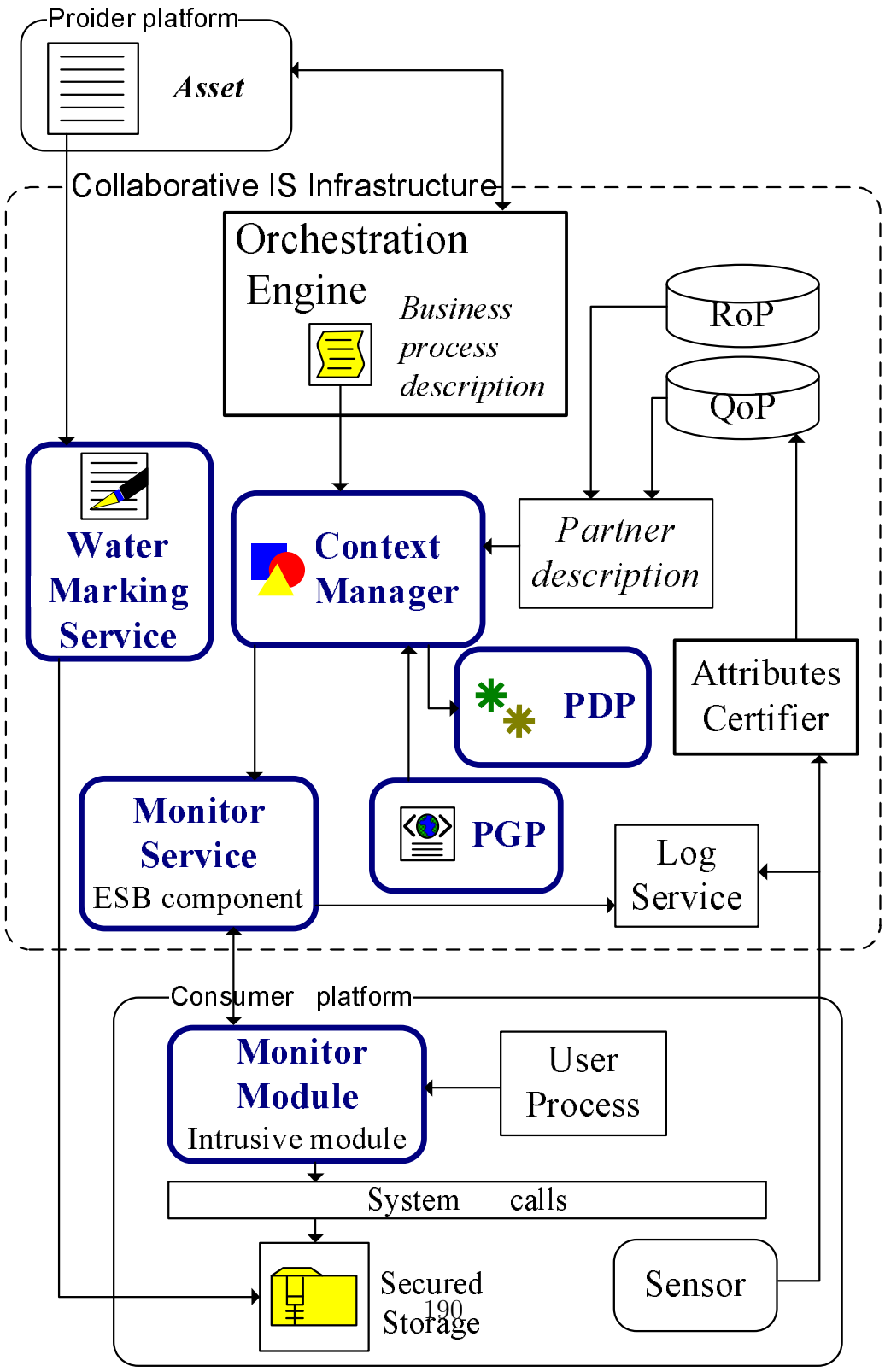


Figure 6.2: Architecture overview

components is given in section 6.4.1.

The '**water marking**' service adds watermarks to provider's data depending on the provider's demand, using state-of-the-art technologies. It can watermark 'multimedia data' [107], 'relation data base' [5] [275] or 'XML files' [333], or data stream [173].

The overall usage control management process is as followings:

- At each step (each interaction of partners) of the business federation, the '**context manager**' analyzes the context status, tracking asset aggregation activities, producing information about which QoP should be negotiated with which RoP, which QoPs should be aggregated and which RoPs should be aggregated. It also collects the providers' policies and consumers' attributes from the policy repository and attribute repository, as these attributes can be provided by a third party ('**Certifier**'), and sends all these information to the **PDP**.
- The **PDP** evaluates the QoP (when implemented with XACML, QoPs should firstly be interpreted to a XACML 'request') with RoP. Then it calls the **PGP**, for aggregating RoPs or QoPs.
- The '**PGP**' has the functionality for detecting potential conflicts between the policies. If there is any conflict, it sends a negative aggregation result to the context manager. Otherwise, it produces the aggregated policies and sends them back to the context manager.
- The '**context manager**' informs the business process engine (or orchestration engine) to halt the business process if the policy negotiation or aggregation is negative. Otherwise it sends the policy evaluation results to the Monitoring Service.
- The '**monitor service**' inspects the usage activities and updates the log service.
- The '**log service**' collects the events in the collaborative context (usage events, events related to the business session and other events in the participant systems). These events also serve as information for updating partners' attributes.
- The '**water marking service**' can be used on provider's demand. In scenarios where the provider's data are used in their original form (e.g. data set, picture, multimedia, E-maps, etc), watermarking can be used as forensic evidence.
- The '**sensor**' represents the mechanism that the 'certifier' uses to inspect the attributes changes that are not caused by usage activities or

context events (otherwise, the log service can not be aware of these changes). The 'sensor' mechanism closely depends on the infrastructure of an application. By now, there is no out-of-the-box technology for it. Usually it should be done with human intervention.

The following sections further describe these components.

6.2 Information flow management in WS-BPEL business process

The 'context manager' (see figure 6.3) provides 'security data provenance' management, i.e. tracking the derivation history of a piece of data.

The context manager cooperates with the business engine to get the description of business process (usually a script of the business process, e.g. the WS-BPEL script). For this, the context manager analyzes the business process before the 'orchestration engine' starts it. The overall procedure of context management is defined as (figure 6.4):

- step 1: The 'context analyzer' loads a business process defined with WS-BPEL and fetches the WSDL files of the business partners defined in the WS-BPEL.
- step 2: The 'context analyzer' uses 'context slicing' method to trace the assets provided by partners in the business process and categorizes sub contexts and allocates the assets to them, according to the assets merging/ inheriting relation. This process generates 'provenance indicator' data, indicating which O-Assets / C-Assets are used to set this C-Asset.
- step 3: For each 'sub context', the 'policies assembler' fetches the RoP and QoP policies of all partners, indicated by their WSDLs. It parses the QoPs of partners' and fetches the attributes, in order to set the requests. Then it sends requests and RoPs to the 'Negotiation and Aggregation' engine ('PDP' for negotiation, 'PGP' for aggregation, as described in section 6.3).
- step 4: If the negotiation and aggregation results for all the 'sub contexts' are positive, the 'context manager' will call the 'orchestration engine' to start the business process.

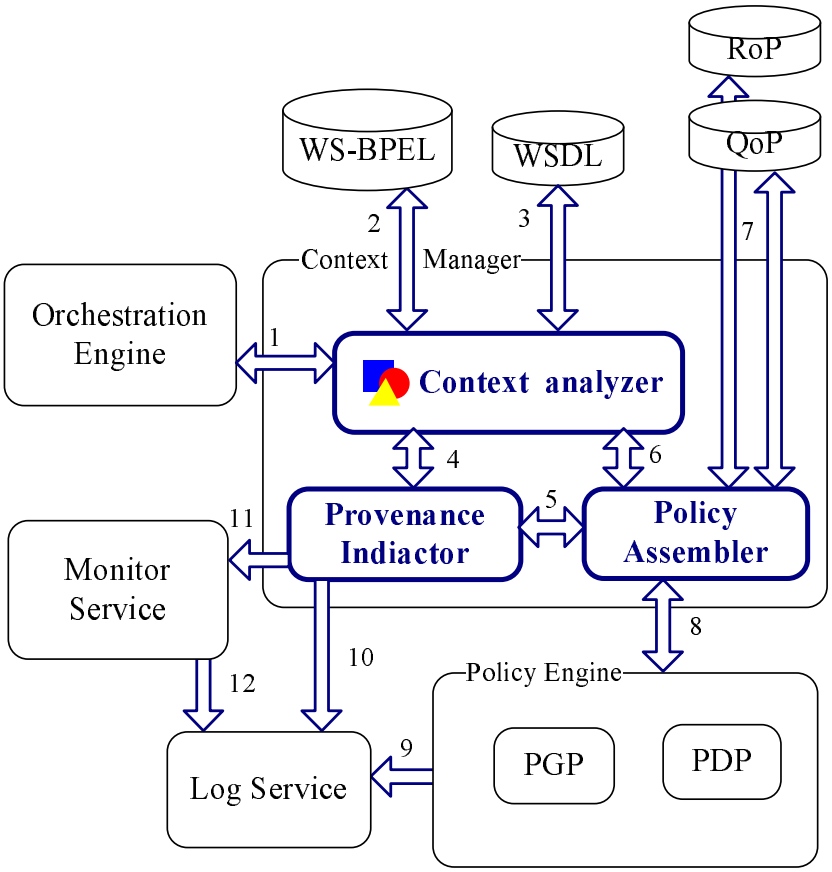


Figure 6.3: Components of context manager

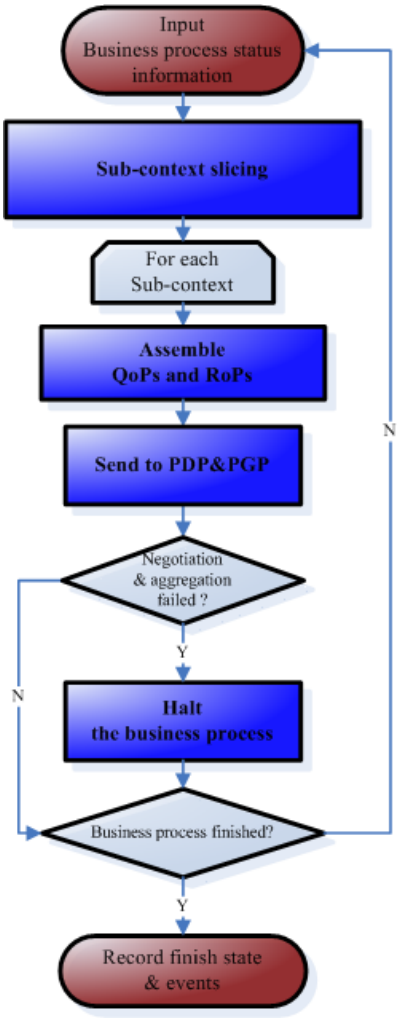


Figure 6.4: Overall procedure of context management

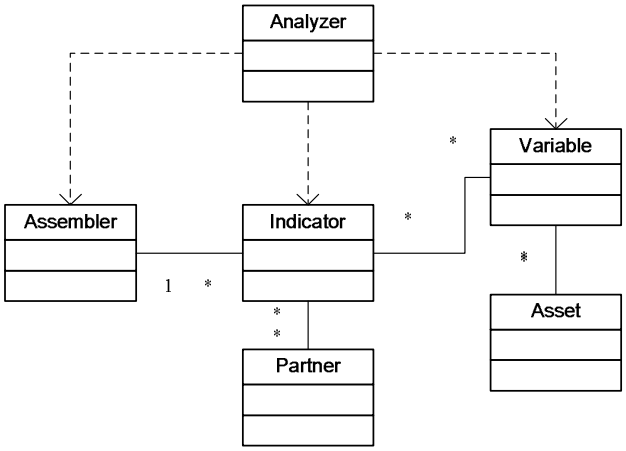


Figure 6.5: Class diagram of context manager

Figure (6.5) represents the main classes implementing these functionalities:

- The 'Indicator' class maintains the 'provenance' data, which is the 'context development tuple' introduced in chapter 5. It has fields indicating 'context name', 'context version', 'parent context', 'asset list', 'step info', 'variable', 'partner' and 'children context'. This data structure is used by the analyzing algorithm (implemented in the 'Analyzer' class) for tracking the context development.
- The 'Analyzer' implements the central functionality of the manager. It analyzes the business process description according to the 'asset based strategy' proposed in chapter 5.
- The 'Assembler' provides methods for adding policies into the contexts. As an output it builds a configuration file that provides information for the PDP and PGP engines (which QoP should be negotiated with which RoP, which QoPs should be aggregated and which RoPs should be aggregated, etc.).

The analyzing algorithm is organized depending on the '$< activities >$' elements in WS-BPEL document. These *activities* can be categorized into several types according to their different impacts on context management:

- **category 1**: The basic *activities* that directly control the interactions between partners ('partnerLink' element) or the value assignment between parameters ('variables' element), including $< receive >$,

$< reply >$, $< invoke >$, $< assign >$, $< invoke >$ and $< exit >$. This kind of *activities* can only result in the create/update/merge/end of context (see chapter 3 for definitions).

- **category 2**: The 'flow control' activity (*activities* that control the workflow) that can lead to context split. This includes $< sequence >$, $< flow >$, $< forEach >$ and $< if >$. For example, in $< sequence >$ structure, if the value (i.e. an 'information' asset) carried by a variable is assigned to two other variables, it may cause context split (because the values assigned to the two variables can be developed differently). Other structures leading to context split includes:
  - a $< flow >$ with a 'parallel' factor that has value '*yes*'.
  - a $< forEach >$ with a 'parallel' factor that has value '*yes*'.
  - an $< if >$ structure involving different *activities* in its branches.
- **category 3**: The 'flow control' *activities* $< pick >$, $< scope >$, $< while >$ and $< repeatUntil >$ by themselves don't lead to context split. Nevertheless, as they can have any other *activities* included in their scope, their children *activities* should be examined.
- **category 4**: The *activities* that are not related to context slicing include $< throw >$, $< wait >$, $< empty >$, $< compensate >$, $< compensateScope >$, $< rethrow >$, $< validate >$, $< extensionActivity >$. Consequently, they are ignored during context slicing.

The context slicing process is described in 'algorithm 1' (associated to the 'coordinator' method in 'Analyzer' Class). It locates the process starting point, i.e. a '$< receive >$', '$< pick >$' or '$< onEvent >$' activity that has a '*createInstance*' factor. Then it creates the first context. After that, it gets all the following *activities* and sends them to the 'analyze' method for tracking asset aggregations. The result is recorded as a 'list' of 'sub contexts' (represented by 'Indicator' objects). Then each 'sub context' is enriched with the '*QoP*' and '*RoP*' of the partners involved in it. Lastly, a configuration file (generated from 'Assembler' object) is generated from the list of sub contexts.

The context slicing method (implemented with 'analyze' method) is described in 'algorithm 2'. It deals with each activity according to its impact to context development (i.e. which category it belongs to). For the basic activity (i.e. those in category 1), it calls a'development' method, which updates the *context* list according to the '*slicer*' factor (creating a new

Algorithm 1 method 'coordinator'

Require:

The business process defined with WS-BPEL

Ensure:

An 'assembler' Object, which comprises the context slicing information;

1: Locate the starting activity of business process
2: Create the first '*asset*' object according to the starting activity
3: Create the first '*context*' object with the first '*asset*' object, the '*variable*' and '*step*' information
4: Create the list of '*context*', which consists only the current *context*
5: Set the value of '*slicer*' to *update*
6: **for** Each *activity* that follows starting activity **do**
7: Call method 'analyze' with the *activity*, the '*slicer*' and the list of *context*
8: Get the updated list of *context*
9: **end for**
10: create 'assembler' object with the list of *context*
11: Enrich the *context* in the list with 'RoPs' and 'QoPs'
12: Output the configuration file with information in the list of *context*

'*context*' if *slicer* =′ *split*′, updating the version of an existing *context* if *slicer* =′ *update*′) to generate a new 'sub context'. Activities of other categories are complex *activities*, so the method checks their children *activities* (by recursively call itself with these *activities* as parameters). When an activity belongs to category '2', the flag '*slicer*' is set to '*split*', so that a new context is created (as siblings of each other) at each recursive call.

6.3 XACML-based policy negotiation and aggregation

A typical authorization system includes a policy decision point (PDP) and a policy enforcement point (PEP). PDP is the point where policy decisions are made. PEP is the point where the policy decisions are actually enforced [316]. We extend the PDP with a PGP to support policy aggregation, forming the 'Negotiation and Aggregation engine' (see figure 6.6).

The engine consolidates the functionality (see 'algorithm 3') for choos-

Algorithm 2 method 'analyze'

Require:

A list of the *Indicator* object, *slicer* and current *activity*

Ensure:

An updated list of the *Indicator* object, *slicer*;

```
1: if activity in 'category 1' then
2:   Call method 'develop' with this activity, the 'context' list and slicer =' update'
3: else
4:   if (activity in 'category 3') then
5:     for Each child activity do
6:       Call method 'analyze' with the activity, the 'context' list and slicer =' update'
7:     end for
8:   end if
9: else
10:  if (activity in 'category 2') then
11:    for Each child activity do
12:      Call method 'analyze' with the activity, the list of context and slicer =' split'
13:    end for
14:  end if
15: end if
16: Output the context list
```

ing one service from a set of services (called 'former services' as they are 'up-stream' information providers) that can provide information for another service (called 'current service', as it calls the 'up-stream' service for information), including the following steps (see figure 6.7):

- '**negotiation**' (statement '1' in 'algorithm 3'): It's achieved between the QoP of the current service and the RoPs of the former services.
- '**QoP aggregation**' (statements '3'-'10' in 'algorithm 3'): It's achieved using services with a positive negotiation result. It detects potential conflicts between the former services' QoPs and the QoP of current service.
- '**RoP aggregation**' (statements '11'-'17' in 'algorithm 3'): It's achieved

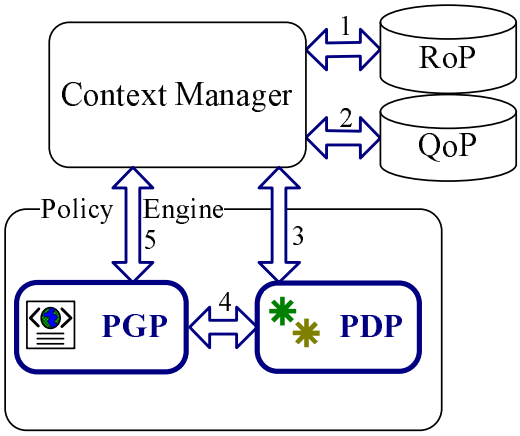


Figure 6.6: Extended Policy Decision Point

with the former services which QoPs are compatible with the QoP of current service. It also detects potential conflict between their RoPs and the RoP of the current service.

- '**recommendation**' (statements '18' and '19' in 'algorithm 3'): It orders the 'former services that are selected by the **negotiation**, **QoP aggregation** and **RoP aggregation** processes, to recommend the services that fit the best the context.

This process is carried out through functionalities provided by classes 'Evaluation', 'NegotiationPDP', 'AggregationPDP', 'Function', 'Aggflag' and 'AggregationResult' (see figure 6.8).

The 'Evaluation' class coordinates the negotiation process (carried out by 'NegotiationPDP') and aggregation process (carried out by 'AggregationPDP'). Each result of negotiation or aggregation is represented by a 'flag' (an instance of 'Aggflag' Class). These flags are used to generate recommendation (by the 'AggregationResult' class). The I/O operation and attribute predicate comparison operation are necessary during the negotiation and aggregation processes (it's provided by the 'Function' class).

6.3.1 Negotiation process

The negotiation process is described in 'algorithm 4' (it starts with the 'negotiate' method, which belongs to 'Evaluate' class). It parses the QoP (to generate 'requests') before sending all the requests and RoPs to the method

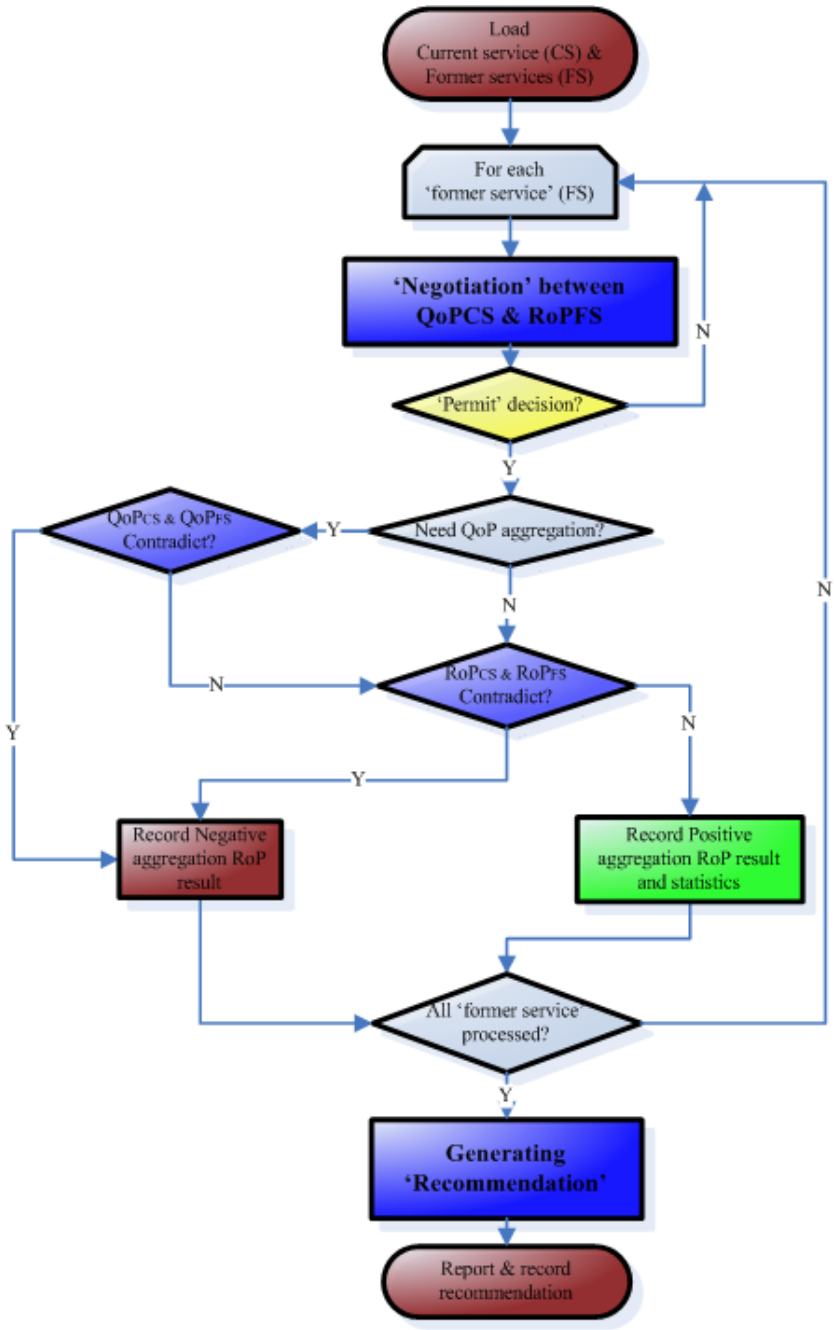


Figure 6.7: Overall procedure of policy negotiation and aggregation

Algorithm 3 Negotiation and aggregation process

Require:

The $confFile$ recording 'current service' and 'former services'

Ensure:

The 'former service' best compatible with 'current service';

```
1: Negotiation between QoP of current and every RoPs of former services
2: if There is positive result then
3:   if There is need of QoP aggregation then
4:     for Each former service in the positive result set do
5:       Aggregation between QoP of current service and the QoPs of the
         former services
6:     end for
7:     Record the positive result set.
8:   end if
9:   for Each former service in the positive result set do
10:    Aggregation between RoP of current service and the RoPs of the
       former services
11:  end for
12:  Record the positive result set.
13: end if
14: Sort the final positive result set and to generate 'recommendation'
```

negotiation, which implements a standard XACML 'Policy Decision Point' (PDP) using 'SUN-XACML-Engine' package. Each 'RoP' is negotiated with all the 'requests'. Positive result are recorded.

6.3.2 Aggregation process

This process (implemented by 'AggregationPDP' class) includes two sub-functions. First, all the 'former services' that permit the request from 'current service' (recorded in an array 'permitFormerSvcs') are sent to a method ('aggQoP') that detects conflicts between QoPs of 'current service' and 'former services' (see figure 6.9). The positive results (recorded in an array 'permitQoPFormerSvcs') are sent to a method ('aggRoP') that detects conflict between RoPs of 'current service' and 'former services' (see figure 6.12). The positive results (recorded in an array 'CompatibleFormerService') represent the 'former services' that can join the collaborative context, as such

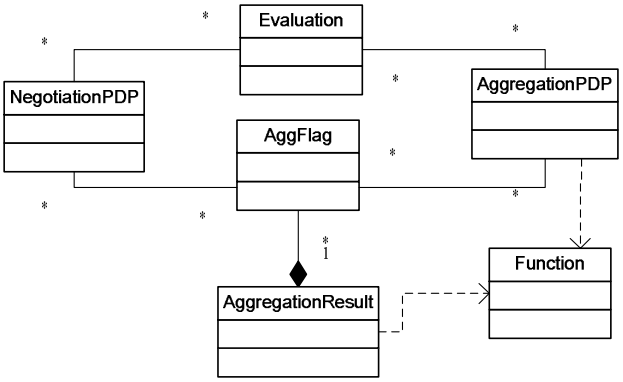


Figure 6.8: Class diagram for PDP and PGP implementation

'former services' permit the request from 'current service' and have QoP and RoP compatible with the QoP and RoP of 'current service' respectively.

The QoP aggregation process is described in figure 6.9. We can see that it's only when both partners will act as 'consumers' in future steps of the context (this can be decided by 'context manager') that the QoP of current service ('CS') and former service ('FS') will be compared. When there is no conflict, their QoPs are aggregated. A statistic is also generated, in order to generate recommendation.

The RoP aggregation process is described in figure 6.10. Similar to the 'QoP aggregation' process, the RoPs of the 'CS' and 'FS' will be compared if they will both provide assets.

These two aggregation processes rely on a method (provided by 'Function' class) to detect conflicts between rules. As defined in chapter 5, two attributes predicates having the same 'AttributeID' and separate 'Attribute Value' scope are conflictable. For example if two attributes have attribute ID 'possess certificate' but unshared value scope definition: 'May, 10-May, 30' and 'June, 2-June, 20', they are contradictable. Our implementation support the comparisons of basic data types according to the requirements of XACML specification (i.e. 'STRING', 'INTEGER', 'DOUBLE', BOOLEAN', 'DATE', 'TIME', 'DATETIME', 'DATETIMEDURATION', 'YEARMONTHDURATION', 'ANYURI', 'X500NAME', 'RFC822NAME', 'HEXBINARY', etc.). A knowledge base can also be included to compare two concepts defined in an OWL ontology.

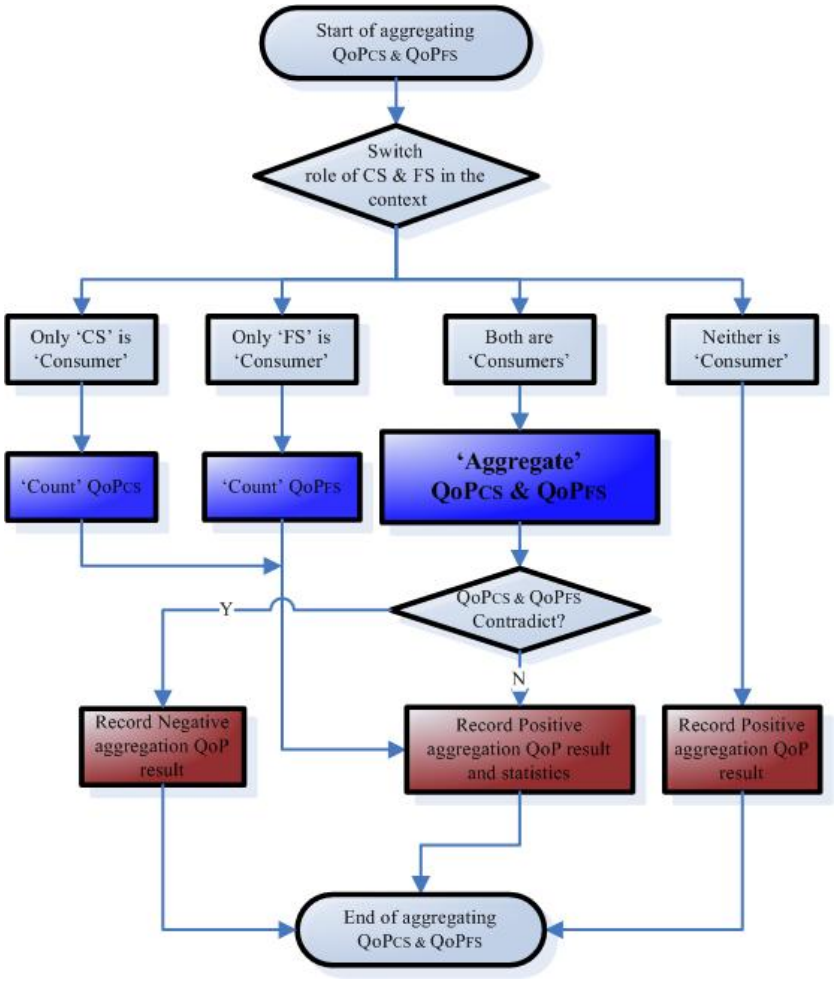


Figure 6.9: QoP aggregation process

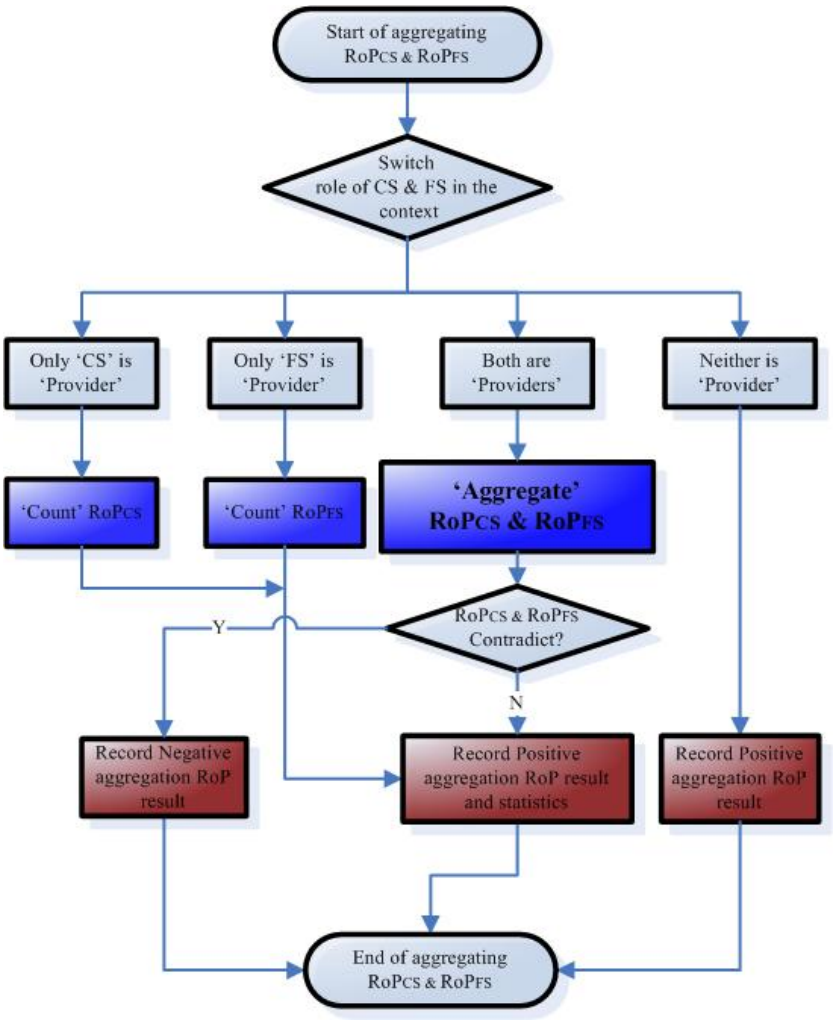


Figure 6.10: RoP aggregation process

Algorithm 4 Method 'negotiation' in class 'Evaluation'

Require:
The $currentQoP$
The list of all $formerRoP$;
Ensure:
The $permitFormerSvcs$, the list of former services that give positive result by negotiation between $currentQoP$ and $formerRoP$;

```
1: Perse the currentQoP, get all the request;
2: for Each formerRoP do
3:   for Each formerRoP do
4:     Call method negotiation in class NegotiationPDP with request and
       formerRoP;
5:     Parse the Result in returned response;
6:     if Result is negative then
7:       break;
8:     end if
9:   end for
10:  if Result is positive then
11:    Index the former service to permitFormerSvcs;
12:  end if
13: end for
14: return permitFormerSvcs;
```

6.3.3 Generating recommendation

The aggregation results between the 'current service' and all the 'former services' are maintained by a class 'AggregationResult'. Providers are ranked (using the method 'getDecision') with the **'principle' of aggregation function** introduced in chapter 5.

This statistic information is encoded as XML file by method 'encode'. The aggregated QoP and RoP are recorded by the field 'resultDoc' of 'AggFlags' class.

6.4 Enforcement of usage control policy

Usage control enforcement requires 'monitoring' mechanisms to inspect consumer's activity and content securing mechanisms to protect asset during

exchange.

6.4.1 Monitoring service

In our policy model, 'right' involves not only the 'usage (consumption)' actions, but also other privileges as 'assume role' and 'delegation'. Such 'non-consumption' privileges lead to the partners' attributes changes, rather than the asset status changes. Usage control system should monitor both the 'usage rights' actions and 'non-usage rights' actions.

6.4.1.1 Usage rights monitoring

The 'usage monitoring' is implemented as an intrusive module (see figure 6.11) provided by trusted third party that resides in the consumer system. It supervises the consumers' consumption activities upon assets. If any activity that is not allowed by provider policies occurs, the monitor alerts the provider.

This intrusive module consists in two components, namely 'port monitor' and 'system call monitor', and a data structure 'system call list'. The 'system call list' is generated from provider policies. It can be a 'white list' of the system calls that represent the allowed usage activities, or a 'black list' including the system calls that can lead a data to a state denied by policy. It is used by the two components when monitoring consumption activities. Major steps of monitoring are as follows:

- When the provider sends a message that carries assets to consumer, the 'port monitoring' starts-up the 'system call monitor'. It extracts the 'asset list' according to provider's policy and sends it to 'system call monitor'.
- The 'system call monitor' inspects the system calls generated by the consumer system (e.g. a Web Service). It tracks the state of all the copies of the asset. It also logs the system calls upon all the copies.
- The 'system call monitor' compares the system calls on all the copies of the asset with the 'system call list'. If an unallowable system call occurs, it sends an alert to the 'context manager' and log the information.
- The 'context manager' will then update the attributes of the consumer, reducing its trust assessment (this step is not illustrated in the figure).

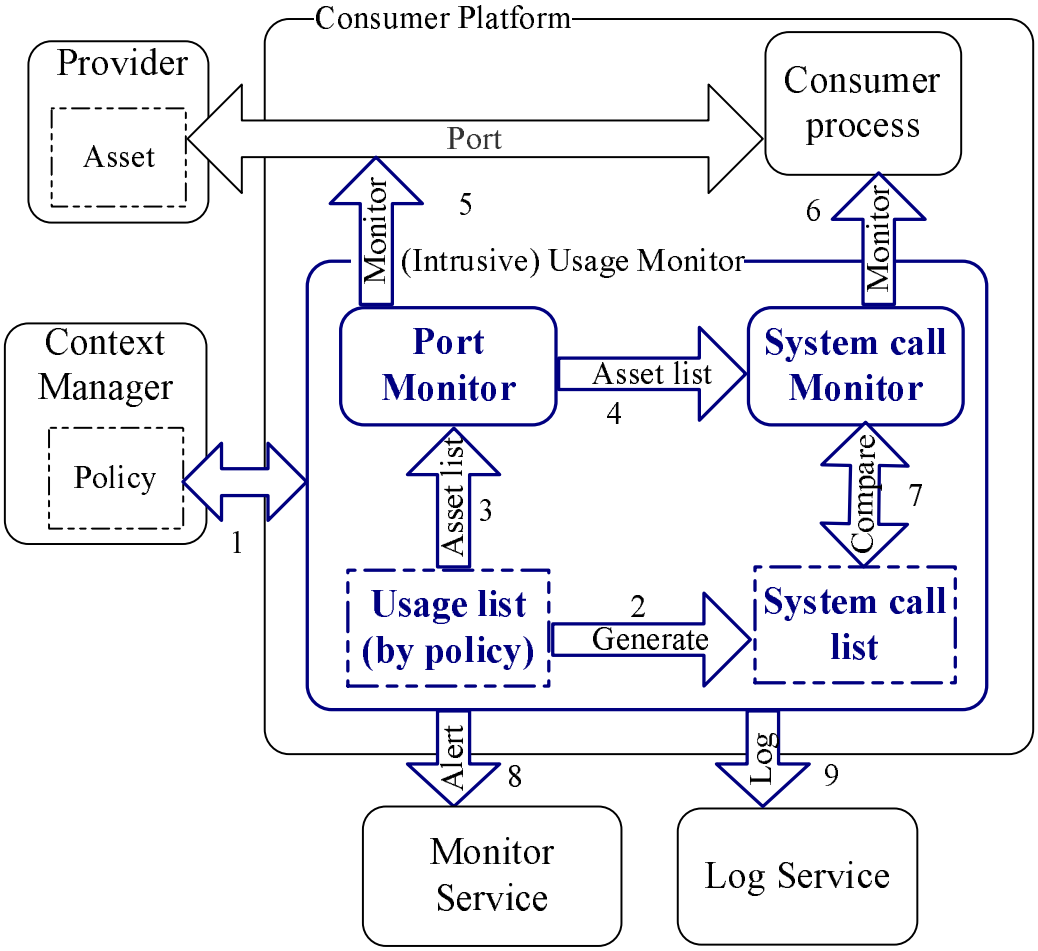


Figure 6.11: Intrusive usage monitor module

- The log of the system calls upon all the copies of the asset is also sent back to the 'context manager'. It updates the context 'provenance' and sends it to the 'logging service'.

In short, the intrusive usage monitor module confines the operations upon assets to the range defined in the policy. For example, it can limit the I/O operations so that the data exchange is between the provider and the consumer. It also makes sure (when required) that the consumer system erases the data in memory and deletes the data stored on the disk at the end of the transaction.

Another important issue is whether the system calls are "original" or not. If some system calls are modified without notifying the monitor, the monitor's ability is compromised. To ensure that only original system calls are achieved, one can leverage the 'trusted bootstrapping' technology [239] [240]. It bootstraps trust in a computer using secure hardware mechanisms (e.g. Trusted Platform Module (TPM) [120]), to monitor and report on the platform software status.

6.4.1.2 Non-usage rights monitoring

The 'assume local role' and 'delegation' rights are 'non-usage' right, which means that granting such rights doesn't directly lead to consumption activities. 'Assume local role' gives a partner the privilege of a 'role' defined in the local system of a party. By this way the partner can act as a member of the local system. By using 'delegation of right', a party gives a partner the privilege to exert a right or to pass the right down. By 'delegation of control', a party gives a partner even the ownership of the asset. Using 'assume local role'/'delegation of right' involves no assets exchange, but the partners' attributes should be updated. 'delegation of control' involves both updating attributes and exchanging assets.

Therefore, a non-usage rights monitor at the context level (e.g. implemented as a ESB component) is necessary for attributes updating (see figure 6.12)

The non-usage rights monitor works with the following major steps:

- When consumer requests a privilege like 'assume local role' or 'delegation', the 'context level monitor module' forwards the request to the context manager.

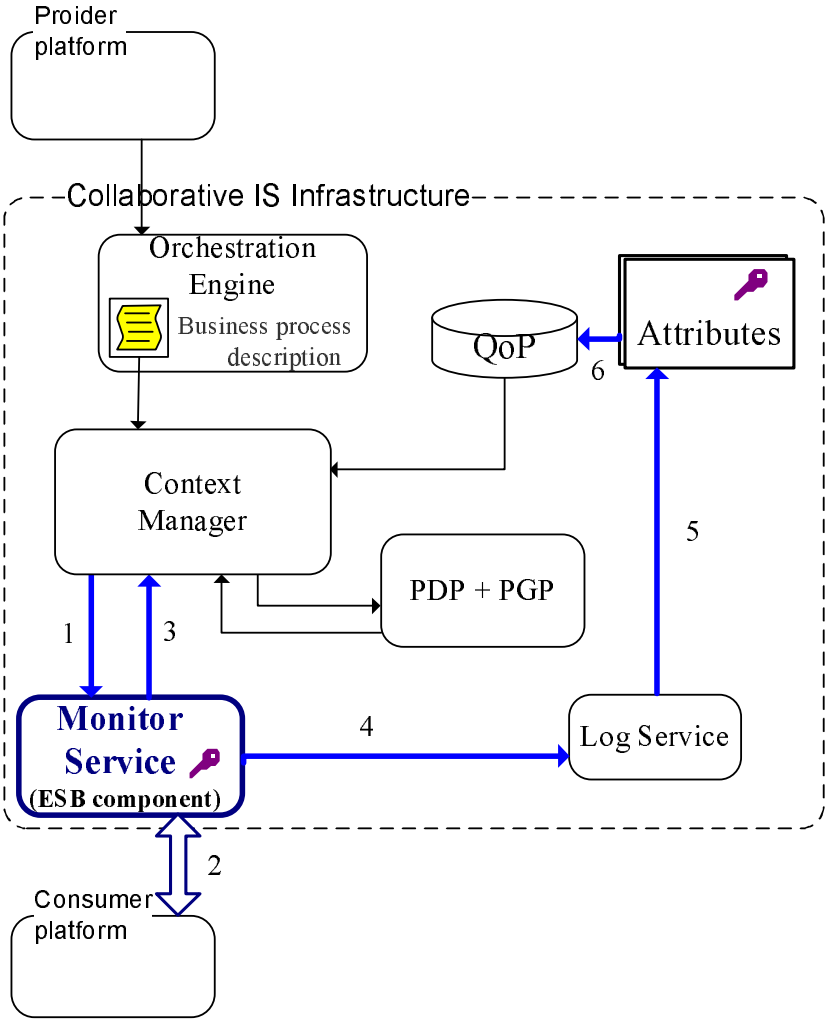


Figure 6.12: Context level monitor module: *Blue lines represent it interactions with other components*

- If the 'monitor module' receives a positive response from the context manager, it sends update information to the 'log service' (this information denotes that the consumer has the privilege).
- The 'log service' updates the attributes of the partner with the granted privilege.
- The privilege is updated in the consumer's QoP.

The 'context level monitor module' can be implemented as a service in the collaboration management system. For example, in Service Oriented Computing, it can be implemented as a Enterprise Service Bus (ESB) component.

6.4.2 Content securing

Content preparing technologies based on encryptions are popular means for securing digital contents. Besides, 'watermarking' technology can be used to identify the re-distribution versions, in order to deal with piracy. A usage control system can involve a 'data watermarking service' This service provider should be deemed as trustful for the participants. Otherwise, usage monitoring mechanism should be imposed upon it.

If the asset sent to the consumer system should be kept for a while, a secured storage may be required by the provider. A trusted storage can be provided at the software level, e.g. by software encryption in the 'intrusive usage monitor module', or at the hardware level, using TPM [120].

6.4.3 Conclusion

It can be seen that our 'right enforcement mechanism' comprises several components, in order to fit to collaborative context and adapt to different usage scenarios:

- 'Usage monitor' takes care of 'consumption activity' at consumer system.
- 'Non-usage' rights (partner privileges and attributes changes) are monitored at ESB level.
- 'Secured content storage' fits to long-term usage, when data should be kept at consumer side for a while.
- In case a provider's data is used always in its original form, e.g. E-map, GIS data, etc., 'watermarking' can be used as forensic evidences, namely for tracking piracy.

There is no usage control system that provides all these components by now. We can find several works that discuss some technologies for these components, which can be used to implement our rights enforcement components:

- [129] [131] discuss 'usage control' enforcement at Operation System, inspecting system calls for monitor usage actions upon information.
- ESB level 'non-usage' rights monitoring can use methods like those in [111] [112], to support secured information exchanging and partners' attributes updating.
- TPM can be used for secured content storage [134].
- Recent years have seen the development of watermarking technologies for relational database [275] [6], XML document [333], XML stream [173], etc. They can be used according to the types of the data that providers want to protect.

6.5 Extending usage policy management

In the following we discuss the features of some components related to the management of policy, attribute and knowledge base, in order to fit to collaborative context.

6.5.1 Policy management

A policy is usually attached to the information resources to which it applies (see [224], [242] and [280] for discussion about this principle). In our system, a party's policy is attached to its O-Asset. The CSP of a collaboration context or its sub contexts is attached to the C-Asset.

On one hand, each party's policy is published to a policy repository and can be updated by the party itself. If the party is an organization, the policy is a unified 'portal' for regulating cooperation with the external world and is under the 'centralized' control of the authority of that party. Besides the policy owners, aggregation engines can also read their policies. The policy repository should also provide protection to parties' policies, being able to prohibit others parties from reading a party's policy (according to the requirements of this party).

On the other hand, a collaboration context security policy (CSP) is managed in a "locally centralized" way by the aggregation engine that creates it

('locally' is used to contrast with 'globally'. As there may be more than one policy repository and more than one aggregation engine in an open ecosystem of services context, e.g. Web Service.).

A participant's policy is defined with 'version': updating the policy will 'create' a 'new version' of the policy. The former version policy will be marked as 'substituted by "new version" at a "time point"'. When a version of the policy is used in a transaction, the owner of that policy should always support this version until the end of the transaction.

6.5.2 Knowledge base management

Knowledge bases include the vocabularies to express domain knowledge. There are several types of knowledge base, used in different layer of a collaborative context:

- **Public-abstract level**
  This is the most general level knowledge base, collecting factors that can be encountered in every application. It usually integrates the most general IT security concepts. The 'vocabulary base' defined in chapter 4 is an example of such a knowledge base.
- **Industry-specific level**
  It extends the 'Public-abstract level' vocabulary base with factors related to different application domains. Industry-specific level vocabulary base is more pertaining to a specific application. For example, for an environmental monitoring system, we may want to incorporate factors like 'temperature', 'wind speed', etc.
- **Local level**
  One party can use its local knowledge base to define its specific factors, e.g. local roles.
- **Collaboration level**
  A collaboration process may produce some information that should be recorded and may be referenced in the future,e.g. by transaction-related events.

Partners in a transaction should have consensus about the industry-specific level knowledge base they use. If a partner wants to modify this knowledge base, it makes a proposition and all the partners 'vote' upon it to decide if the modification will be carried out or not.

6.5.3 Attribute management

We define an 'attribute repository (AR)' to enhance the trustworthiness and prevent attack scenarios (like when an attribute owner deliberately hold-back some attributes, e.g. its history-actions to 'bypass' the 'separation-of-duty' rules in order to gain rights it should not have). Attributes can be certified by CAs to increase the trustworthiness upon them (like the identity certification mechanism).

In a collaboration context participants' attributes can be mutable, as discussed in UCON [238]. In our model, the OAT, SAT and CNAT can all be mutable. We specify two kinds of mutation.

- The '**transaction-driven mutation**' is triggered by execution of rights, Session Events, or Security-Event. As discussed in UCON model, there are the '**pre-change**', '**ongoing-change**', '**post-change**' which happen before, during and after the happening of an event separately. The 'ongoing-change' of an attribute will highly impact concurrent control.

- The '**externally-caused mutation**' are changes that are not related to a transaction, i.e. '**consumer-spontaneous change**', '**provider-spontaneous change**' and '**environment prompted change**'.

6.5.4 Conclusion

To summarize, the main features of our extended policy management components are:

- Policy (file) protection and updating mechanism are designed considering the collaboration context.
- Knowledge base is organized in an extensible structure. Users can modify it according to requirements in their application domains.
- Attribute management mechanism emphasizes on the trustworthiness of partners' attributes, prohibiting deliberate conceal.

It can be seen that the usage control enforcement requires several components. Due to time limits, we have implemented only some of them. The performance test results are presented in the following section.

6.6 Benchmarking

In order to evaluate our propositions and implemented components, we set a benchmark that focus on:

- the 'policy scheme level' features as:
  - the policy language that can express 'usage' rights and security-related attributes;
  - the mechanism to deal with policy aggregation, fitting to assets aggregation scenarios;
  - the context management method coping with complex collaborative business processes;
  - the vocabulary base including collaborative context security factors, in order to facilitate policy authoring.
- the 'implementation level' features as:
  - engines that support policy negotiation and aggregation;
  - components for policy enforcement, e.g. usage monitor, content securing, policy management.

There are several efforts aiming at building usage / access control system for collaborative/decentralized contexts in these latest years. We use above criteria to compare these works with our proposition.

6.6.1 Comparison with other 'usage control' systems

We provide a brief analysis of recent access control systems and see how they match the criteria we propose.

Table 6.1 represents the policy scheme level features of these systems. As it can be seen from the 'policy model' column, many works are based on 'usage control' (UCON, C-UCON) model (No. 1-10 in table 6.1). Several of them are 'collaborative' (C-UCON, No. 5-10) schemes, possessing a 'policy aggregation' method (those denoted '**y**' in the column '**aggregation**') or a 'context management' method (those denoted '**y**' in the column '**context management**'). Very few among these systems possess vocabulary bases.

Table 6.2 represents the implementation level features. We can see (from column '**language**' of table 6.2) that the UCON policies are implemented mainly in XACML or semantic Languages (e.g. RDF). But very few among

No.	system	policy model	aggregation	context manage	vocabulary
1	$UCON_{ABC}$ [238]	UCON	n	n	n
2	DOCOMO [131]	UCON	n	n	y
3	[134]	UCON	n	n	n
4	[66]	UCON	n	n	n
5	xfACL [215]	C-UCON	y	n	n
6	FORM [265]	C-UCON	n	y	n
7	[259] [258]	C-UCON	y	n	y
8	[268]	C-UCON	n	y	n
9	[127]	C-UCON	n	n	n
10	[20]	C-UCON	y	n	n
11	[219]	access control	y	n	n
12	[75] [74]	access control	y	n	y
13	[162]	access control	y	n	n
14	[135]	access control	y	n	n
15	[48]	trust	y	y	n
16	[285]	obligation	y	y	n
17	[284]	trust	n	y	n
18	[100]	propagation	n	y	n
19	[97]	access control	n	n	y
20	AIR [151]	access control	n	y	y
21	[187]	access control	n	y	n
22	[273] [274]	access control	n	y	n
23	[47]	IFC	n	y	n
24	[283] [178] [31]	QoS	n	y	n
25	[33]	WS-Security	y	y	n
26	[281]	access control	n	y	n
27	our work	C-UCON	y	y	y

Table 6.1: Features of some collaborative access control systems: part A

No.	language	engines	enforce-ment	test	main feature
1	XACML	n	LDAP	y	UCON
2	XACML	n	sysCall	n	UCON
3	n	n	TPM	n	enforcement with TPM
4	POLPA	Globus	JVM/ network	n	JVM level enforcement
5	XACML	Fsharp	n	n	Fsharp engine
6	n	n	n	n	federated REL
7	SDS	n	n	n	vocabulary base
8	RDF	n	y	n	protocol level
9	n	n	n	n	law in policy
10	n	n	n	n	declassification
11	n	n	n	n	originator control
12	n	n	n	n	role mapping
13	n	n	n	n	role mapping
14	n	n	y	n	weighted role
15	protocol	n	y	y	quantitive certificate path
16	n	n	n	n	quantitive domain security
17	n	n	y	n	quantitive info-flow rate
18	NIMD/ KANI	SHIN	y	n	Semantic Web technology
19	OWL+ XACML	y	y	n	Semantic Web technology
20	RDF	y	y	y	Semantic Web technology
21	n	n	n	n	dessimination
22	n	n	n	n	dessimination
23	n	n	n	n	flow path
24	n	n	AGENT	n	QoS data collection
25	TulaFale	y	n	n	link WS-security policy for WS composition
26	abstract	n	y	n	Event drive
27	XACML	y	y	y	C-UCON, enriched 'rights'

Table 6.2: Features of some collaborative access control systems: part B

these systems presents technical details of negotiation engine design (see those denoted '**n**' in the column '**engine**'). As far as policy enforcement is concerned, mechanisms traverse hardware level (biding to **TPM**), **system call** level, **JVM** level and application level (**LDAP**). Several works propose implementation architecture, but few of them conducts performance test.

It can be seen from table 6.1 and table 6.2 that most of these works match only a small part of the criteria discussed previously, whereas our work match most of them. Our policy model provides 'usage control level' protection. It fits to collaborative context with methods for policy aggregation and context management. A vocabulary set is built with an 'open (distributed / collaborative) systems' viewpoint. At the implementation level, we provide methods for translating our abstract syntax to XACML and discuss the mechanisms of negotiation and aggregation engines, as well as the architecture for policy enforcement. We have developed some components (negotiation engine, aggregation engine and context manager) and tested their performance with some experiments, the results presented in the following section.

6.6.2 Performance analysis

This section presents the experimental results of our policy engines, including the PDP, the PGP and the context manager. These are the central components of our implementation.

6.6.2.1 Policy engines

Testing with 'TPTP' The following figures show the performance of PDP and PGP in terms of execution time and memory consumption, based on experiments with the profiling tool 'Test and Performance Tools Platform' (TPTP) [103].

Figure (6.13) shows the execution time analysis based on the 'C' and 'G' of the 'Collaborative context scenario' in chapter 4, which consists in negotiation between QoP_C and RoP_G and aggregation of QoP_C with QoP_G and RoP_C with RoP_G. In this figure, the class 'engine' is the entrance for the experiment. 'Evaluation' is the main class of the package. It coordinates the negotiation process, which is carried out by class 'NegotiationPDP', and the aggregation process, which is carried out by classes 'AggregationPDP', 'AggregationResult' and 'AggFlags'. It can be seen (from the column 'Base Time') that:

Execution Statistics - Engine at liesp-port1 [PID: 5680]

Package	Base Time (seconds)	Average Base Time (seconds)	<Cumulative Time (seconds)	Calls
[-] (default package)	100,00 %	0,44 %	100,00 %	100,00 %
[+] Engine	6,76 %	6,76 %	100,00 %	0,44 %
[+] Evaluation	8,82 %	1,47 %	93,24 %	2,65 %
[+] Functions	50,52 %	0,35 %	50,52 %	63,27 %
[+] NegotiationPDP	32,82 %	10,94 %	32,82 %	1,33 %
[+] AggregationPDP	0,89 %	0,02 %	10,49 %	17,26 %
[+] AggregationResult	0,04 %	0,01 %	1,30 %	2,21 %
[+] AggFlags	0,15 %	0,01 %	0,15 %	12,83 %

Figure 6.13: Time percentages of components' execution

- The time consumption of the aggregation process is about 1/3 the negotiation process time consumption.
- The aggregation process and negotiation process take about half of the total execution time.
- The other half execution time is taken by 'Functions' class, which consists mainly methods for I/O operation.

Figure (6.14) presents the content of the 'Functions' class. We can see that:

- The I/O operation takes 49% of the total execution time (see methods 'getJdomDoc', 'jdomOutput', 'convertToDOM' and 'creatAttributeNode').
- The methods for searching 'knowledge base' (vocabulary base) only takes 1.5% of the total execution time (see methods 'toFindConradict', 'findContradict', 'entail', 'imply', 'coexist' and 'compatible').

Figure (6.15) shows the memory consumption analysis based on the 'C' and 'G' of the 'Collaborative context scenario' in chapter 4. We can see that:

- The aggregation functionality (see 'default package' and 'javax.xml.parsers') only takes about 18% of the total memory consumption.
- The negotiation functionality (other packages, they are all included to 'SUN XACML Implementation' package) takes about 82% of the total memory consumption.

Execution Statistics - Engine at liesp-port1 [PID: 5680]

Package	Base Time (seconds)
[-] (default package)	100,00 %
[+] NegotiationPDP	32,82 %
[+] Engine	6,76 %
[+] Evaluation	8,82 %
[-] Functions	50,52 %
getJdomDoc(java.lang.String) org.jdom.Document	42,83 %
jdomOutput(org.jdom.Document, java.lang.String) void	1,20 %
convertToDOM(org.jdom.Document) org.w3c.dom.Document	4,74 %
toFindContradict(java.lang.String, java.lang.String, org.jdom.Element, org.jdom.Namespace) boolean	0,91 %
entail(java.lang.String, java.lang.String, org.jdom.Element, org.jdom.Namespace) boolean	0,40 %
findContradict(java.lang.String, java.lang.String, org.jdom.Element, org.jdom.Namespace) boolean	0,07 %
createAttributeNode(org.jdom.Element) com.sun.xacml.attr.AttributeValue	0,20 %
imply(java.lang.String, org.jdom.Element, java.lang.String, org.jdom.Element, org.jdom.Element, org.jdom.Namespace) boolean	0,05 %
coexist(java.lang.String, java.lang.String, org.jdom.Element, org.jdom.Namespace) boolean	0,04 %
compatible(java.lang.String, org.jdom.Element, java.lang.String, org.jdom.Element, org.jdom.Element, org.jdom.Namespace) boolean	0,04 %
[+] AggregationPDP	0,89 %
[+] AggregationResult	0,04 %
[+] AggFlags	0,15 %

Figure 6.14: Detail time percentages of components' execution

Memory Analysis - Engine at liesp-port1 [PID: 3988]

Package	Live Instances	Active Size (bytes)	Total Instances	<Total Size (bytes)
[+] com.sun.xacml.cond	79,81 %	87,71 %	49,45 %	67,04 %
[+] javax.xml.parsers	4,04 %	1,82 %	19,41 %	11,44 %
[+] (default package)	4,35 %	6,65 %	4,95 %	7,33 %
[+] com.sun.xacml.attr	3,11 %	1,19 %	9,89 %	5,84 %
[+] com.sun.xacml	0,00 %	0,00 %	3,66 %	2,75 %
[+] com.sun.xacml.ctx	0,31 %	0,19 %	2,38 %	2,01 %
[+] com.sun.xacml.combine	4,04 %	1,57 %	2,38 %	1,17 %
[+] com.sun.xacml.cond.cluster	0,00 %	0,00 %	4,03 %	1,03 %
[+] com.sun.xacml.attr.proxy	4,35 %	0,88 %	2,56 %	0,65 %
[+] com.sun.xacml.finder	0,00 %	0,00 %	0,73 %	0,51 %
[+] com.sun.xacml.finder.impl	0,00 %	0,00 %	0,55 %	0,23 %

Figure 6.15: Memory consumption of components' execution

The analysis shows that PGP takes 1/3 the time of the PDP and 1/4 the memory consumption of the PDP. Therefore we can say that the aggregation process we add for collaborative scenario doesn't introduce much performance lost.

Deployment testing In the followings we present performance of the PDP and PGP when we deploy them on specific environments.

Figure (6.16) shows the execution time of PDP for the sample policies and sample request provided with the 'SUN XACML Implementation' package. The 'time1' is the based on the environment of 'Intel T7250 (Dual Core 2.0 GHz)' CPU and '1.99 GHZ, 3.49G' RAM. The 'time2' 'Intel T2330 (Dual Core 1.6 GHz)' CPU and '2.00G' RAM (In the following other figures, 'time1' and 'time2' are as the same meaning here). The figure shows 16 samples of the combination of the 4 policies and 4 requests (detail information is in figure 6.17). Each column is labeled with the negotiation result of the sample it represents, 'N' representing 'Negation', 'P' representing 'Permission'. Generally, a negotiation producing 'P' result takes a slightly shorter time than the negotiation producing 'N' result.

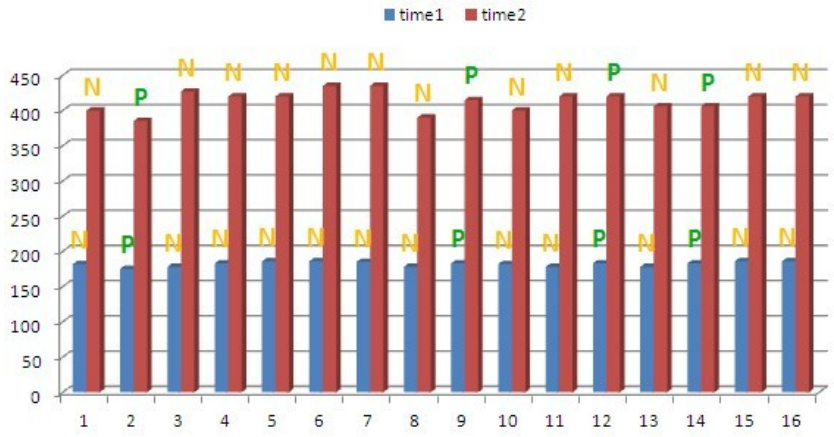


Figure 6.16: PDP performance with Sun's examples

Figure (6.17) gives detail information of the 16 samples in figure (6.16).

The performance of PGP ('AggregationPDP' class) deployed in the two environments is presented in figure (6.18). It shows the time consumption of aggregation depends on the size of the policy in terms of 'number of attribute predicate (ap)' (e.g. '5*5' means that with two policies both having 5 'aps',

No.	request	ap_r	policy	rules	ap_p	result	time1	time2
1	door-access	3	generated	2	7	N	182	400
2	generated	4	generated	2	7	P	175	385
3	resource-content	7	generated	2	7	N	178	427
4	sensitive	4	generated	2	7	N	183	420
5	door-access	3	obligation	2	8	N	186	420
6	generated	4	obligation	2	8	N	186	435
7	resource-content	7	obligation	2	8	N	185	435
8	sensitive	4	obligation	2	8	P	178	390
9	door-access	3	selector	1	4	N	183	415
10	generated	4	selector	1	4	N	182	400
11	resource-content	7	selector	1	4	P	178	420
12	sensitive	4	selector	1	4	N	183	420
13	door-access	3	time-range	3	7	P	178	406
14	generated	4	time-range	3	7	N	183	406
15	resource-content	7	time-range	3	7	N	186	420
16	sensitive	4	time-range	3	7	N	186	420

Figure 6.17: Detail information of negotiation on Sun's examples. *The meanings of column names are:*

- *'request' is the name of the request file;*
- *'ap-r' is the number of 'Attribute predicate' in the request;*
- *'policy' is the name of the policy file;*
- *'rules' is the number of rules in the policy;*
- *'ap-p' is the number of the 'Attribute predicate' of the policy;*
- *'result' corresponds to the results ('N' or 'P') in figure (6.16);*
- *'time1' and 'time2' are the same as those in figure (6.16).*

.

there are $5 * 5 = 25$ 'ap' pairs to process during the aggregation). As we may expect, the statistic of execution time grows with the number of 'ap' pair grows. As shown in figure (6.19), the processing time per 'ap' pair decreases as the size of policies grows. This suggests that the PGP has good scaling property. It deals well with cumbersome policy files, which is probably the case when applying our model to complex scenario for large organizations or organizations with abundant security requirements.

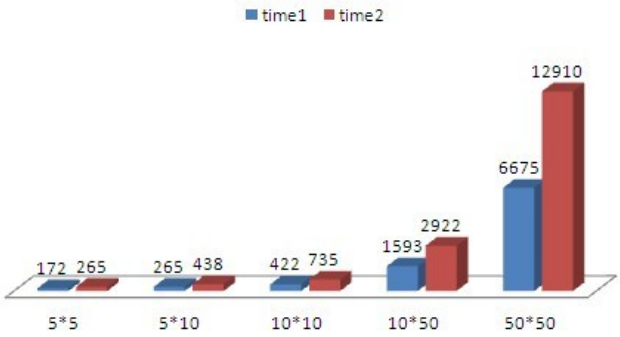


Figure 6.18: PGP performance on different policy sizes

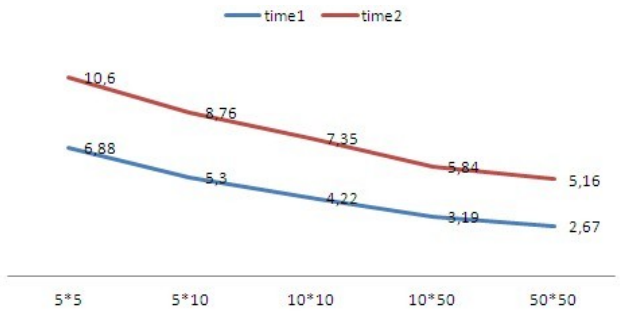
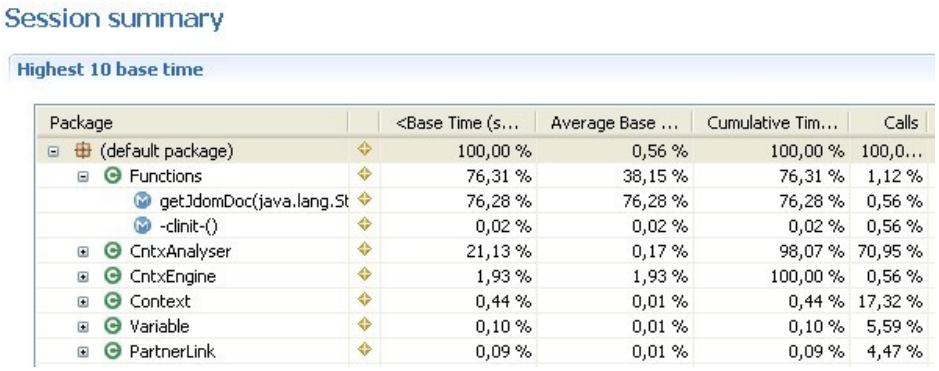


Figure 6.19: PGP performance per 'attribute predicate' pair

6.6.3 Context manager

This section analyzes the performance of the 'Context Slicing' component, based on experiment with the five 'Sample Processes' taken from the WS-BPEL2.0 specification [145] ('initial sample' and other 4 samples in section '15 Examples').

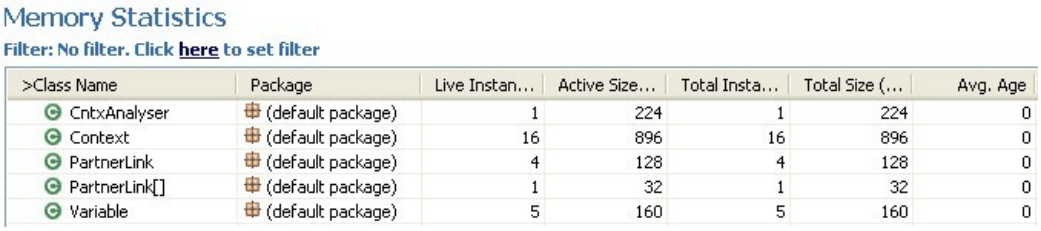
Testing with 'TPTP' Figure (6.20) and (6.21) show the TPTP interface of 'execution time' analysis and 'memory consumption' analysis separately. They are both based on running the package on the 'initial sample'. As shown



Package	<Base Time (s...	Average Base ...	Cumulative Tim...	Calls
(default package)	100,00 %	0,56 %	100,00 %	100,0...
Functions	76,31 %	38,15 %	76,31 %	1,12 %
getJdomDoc(java.lang.St	76,28 %	76,28 %	76,28 %	0,56 %
-clinit-()	0,02 %	0,02 %	0,02 %	0,56 %
CntxAnalyser	21,13 %	0,17 %	98,07 %	70,95 %
CntxEngine	1,93 %	1,93 %	100,00 %	0,56 %
Context	0,44 %	0,01 %	0,44 %	17,32 %
Variable	0,10 %	0,01 %	0,10 %	5,59 %
PartnerLink	0,09 %	0,01 %	0,09 %	4,47 %

Figure 6.20: Time percentage of components' execution

in figure (6.20), the I/O operation (the method 'getJdomDoc') takes 76% of the total time. The analysis process itself only takes 24% the total time. As the file size of a BPEL document is usually very small, the I/O time doesn't change much with different BPEL files. Therefore, we can then conclude that the algorithm scales well with complex (and long) BPEL processes.



>Class Name	Package	Live Instan...	Active Size...	Total Insta...	Total Size (...	Avg. Age
CntxAnalyser	(default package)	1	224	1	224	0
Context	(default package)	16	896	16	896	0
PartnerLink	(default package)	4	128	4	128	0
PartnerLink[]	(default package)	1	32	1	32	0
Variable	(default package)	5	160	5	160	0

Figure 6.21: Memory consumption of components' execution

As shown in figure (6.21), the memory consumption of the context slicing method is insignificant. The 'Total size' of memory consumption by all the instances is 1440 bytes. Therefore, its impact is trivial when being deployed with a orchestration service.

Deployment testing Figure (6.22) shows the performance of deployments on the two environments described previously, in terms of execution time

for processing the 5 sample BPEL processes. Detail information of the 5 processes is shown in figure (6.23).

Figure 6.22: Performance of deployment on different environments

We can see from figure (6.22) that the performance change is mild when deploying on different environment.

Examples	time1	time2	partnerLink	variable	basic_activity
Initial example	156	250	4	5	10
Example1	140	218	1	3	9
Example2	156	242	5	7	18
Example3	140	211	3	3	5
Example4	156	234	3	6	14

Figure 6.23: Detail information of the sample BPEL processes. *Meanings of column names are:*

- *The column 'partnerLink' describes the number of 'partnerLink' element in each BPEL file,*
- *'variable' the number of 'variable' element,*
- *'basic-activity' the number of elements 'receive', 'invoke', 'copy' and 'reply' combining together.*

We can see from figure (6.23) that the 'Initial example' and 'Example2' are more complex BPEL processes. As such their processing time (shown in figure 6.22) are longer.

6.7 Conclusion

This chapter presented our implementation architecture for collaborative usage control. We detailed the mechanisms of the context management component, policy negotiation component, policy aggregation components, as well as the enforcement components (usage monitor, logging service, etc). The performance of context management component, policy negotiation component, policy aggregation components, in terms time and memory consumptions, are presented. The development and performance test of enforcement components will be done in future works.

Chapter 7

Conclusion

End-to-end security aims at protecting an asset (asset here refers to the process, e.g. web service, program, and information, e.g. data, meta-data) from their creation stage to the "demolition". It ensures that the transmission, consumption and propagation are always in the scope defined by provider intention. Our solution relies on the integration of technologies for security management, risk mitigation, policy design, policy analysis, collaborative context management, low-level (OS, VM or hardware) attestation, etc.

The end-to-end security point of view is widely adopted in the research and practice for ensuring features like privacy, regulated information flow, regulated information dissemination and Intellectual Property protection in the application domains like Geography Information System (GIS), Web Service, Social Network, Health-care Information System, Cloud, E-research, Supply Chain, Collaborative Enterprise, etc. In order to provide the life-long and evolving protection, 'end-to-end' security for such federated / collaborative business scenarios should take each partner's definition about the security service level for the assets it provides and the 'usage' rights on the assets.

With the development of knowledge economy and service economy, as well as the technological progress in Information Technology, more and more federated business models are created. This trend is evidenced in the public (e.g. government [82]), private (e.g. corporate [34]) and personal (e.g. social network [48]) sectors. However, the risk of Intellectual Property infringement is the major barrier for parties to move toward such business model. For mitigating such risks, a 'full asset lifecycle' protection is necessary.

7.1 Contributions

For a life-long protection of assets, providers' requirements should be collected, including e.g., which usage is allowed by which consumer (for managing its digital rights), which security service should be provided (mitigating traditional security menaces), etc. To meet such requirements, the consumer side mechanisms to monitor usage activities and collaborative context status are necessary. These mechanisms should cope with assets aggregation, in order to provide life-long protection fitting the collaborative context. Aiming at achieving this goal, this thesis explored several related issues.

1. Based on the analysis of the characteristics of several access control model, as the DAC, MAC, RBAC, OrBAC, ABAC and UCON, we defined a concise syntax for representing the 'basic' usage control policy, which expresses the assets provides 'requirements' (usage policy, similar to REL in DRM [136]).

Specifically, we give a formal semantics that describes the request evaluation process according to a policy in basic UCON policy. This is done using abductive logic programming and event calculus, based on a survey of the conventional formulation methods used in policy definition and analysis. We also present our survey on works using temporal logics to describe the behavior of a system under UCON policy regulation [327] [249]. These regulations capture the general temporal features that are common for UCON system. Thus our 'basic' UCON policy model complies with them.

We provide a method to transfer the abstract UCON language into a XACML implementation, in a straightforward manner. We also build a negotiation engine that implements a XACML PDP, with the 'SUN XACML implementation' package [287], and tested its performance. This engine is added to the 'PEtALS' open source software.

2. We make some extension on the 'basic' usage control model for handling collaborative context. Firstly, the syntax is extended to declare the owner and the lifecycle of the rule, so that the system can identify relevant policies (that is, policies related to the two assets) and aggregate them when assets from two owners merge. To solve this problem, we propose an aggregation method based on an 'Integration Algebra', which extracts the semantics of the reasoning process for co-authorizing a right. Basically, it uses the principles of 'specificity precedence' and 'deny precedence' for the collaborative context as the access to a C-Asset (artifact of the collaborative business process) is refused if the access to one of it sources (O-Assets – the

assets from partners) is refused. We've built an aggregation engine leveraging JAVA DOM and JDOM packages and reported the performance testing results.

3. Our goal is to coordinate security requirements from partners in a collaborative context, using negotiation and aggregation mechanism supported by access control (and usage control) policy models. The ability of describing conventional security factors is critical to manage security attributes of partners. We carry out a survey on factors related to IS security management, privacy and usage control. Based on this, a vocabulary is proposed that collects these factors. Such a vocabulary base characterizes the common security factors related to general collaborative contexts. It enables the collaborative usage control model to be aware of conventional security issues (hardware security, OS level security, people factors, etc). The vocabulary base is built as an ontology supported by semantic web technology. It has the capability to describe enriched relations between concepts. These relations can be taken as 'attributes' in the UCON policy model.

4. To apply on a federated (collaborative) IS context, the system should possess the ability to cope with dynamic business process, being aware of the assets derivation process. Based on the analysis of the characteristics of collaborative business process, we proposed a 'provenance' information management method. It uses an approach similar to System Dependency Graph (SDG) to capture the 'assets derivation' pattern. We gave a detail discussion of the rational for this method and illustrated its working process using two sample use cases (a simple one and a full-fledged one). We've developed an algorithm to analyze business process defined in WS-BPEL. We've made experiments with several WS-BPEL documents and reported the performance testing results.

5. As far as enforcement of usage control policy is concerned, we proposed an architecture and described the functionalities of its components. The main component is the 'monitor service'. It is composed of two modules. The *intrusive module* resides on the consumer platform for monitoring the asset consumption activities, relying (mainly) on inspecting system calls on the consumer platform. The *ESB component* is deployed on the partner's inter-media infrastructure. It is in charge of inspecting the fulfillment of a right that is not a consumption activity on consumer platform, for example the 'assume local role' or 'delegation' rights.

This monitoring service should be provided by a trusted third-party as the consumer and ESB provider can allow it to inspect the operations in

their systems, which may reveal their business secrets. The third-party only reports to the provider the evidence of usage control policy breach. It should not 'leak' any other sensitive information, in order not to damage the intellectual property value of the hosting system.

These works can serve as foundations for technical approach of intellectual property management in collaborative corporation IS. But we can also see several limits.

7.2 Limits

1. A method for analyzing the characteristics of the collaborative IS, in order to collect requirements related to end-to-end security, is the 'starting point' for composing usage control policies. In order to fit the interoperability requirement (with other corporate security works), this method should be based on canonical standards as EBIOS, OCTAVE, ISO27002, SNA, etc. It should take into consideration traits of distributed, parallel and collaborative systems. However, due to time limits, we didn't design a comprehensive method for the security requirements analysis task.

2. A full-fledged system should support friendly human-machine interface for policy authoring, policy administration and log management. This involves the improvement of technologies as information organization and policy visualization, to convenient human-comprehension of issues as policy effects, assets derivation during business process, events related to transaction, etc. Our architecture gives only a brief discussion of this aspect, partly because it's not the central issue of our research, partly due to time limits.

3. The usage control enforcement architecture is not fully implemented. The mechanism of the 'monitor service' is closely related to the platform infrastructure specific to an application domain. For example the monitoring mechanisms at OS level, VM level, network level and ESB level will be greatly different. The enforcement in a particular context involves mapping high-level 'rights' definition to the low-level 'operations' in that platform, namely a 'policy refinement' [330] process. The mechanisms for inspecting the operations, and blocking some of them, are also specific to each application domain.

7.3 Future works

With the development of decentralized collaborative computing systems, e.g. Collaborative Enterprise, Supply Chain coalition, Web Service, P2P, etc, the requirements for protecting partner's information (and process) grow, leading to research progress in fields as *information flow control*, *usage control*, *privacy protection*, etc. The following issues can be studied to overcome the limits we identified and extend our work:

- Developing a comprehensive risk analysis and security management method for collaborative context, based on study of conventional methods as those in EBIOS, OCTAVE, ISO21002, etc.
- Developing policy authoring system, based on constrained natural language, using 'SWRL+Protege+SBVR' or using methods mentioned in [154].
- Developing policy visualization system, using methods as those in [253] [291] [153] and considering other possible technologies.
- Providing more comprehensive Policy Ratification system, like 'EXAM' [188].
- Specifying more detailed Policy Refinement for mapping high level policy to low-level operation in, e.g. in Virtual Machine (VM) level, Operating System (OS) level, Mobile/Pervasive Computing, etc.
- Developing more feasible monitoring mechanism for IS, based on studying 'rootkit' and information track for VM, OS or Pervasive systems.
- Incorporating QoS factors to the security knowledge base, enabling the policy model to be applied in 'Qos' management.
- Implementation the policy model with Semantic Web, e.g. OWL+SWRL technology, improving flexibility.

The 'end-to-end' security for collaborative context is a rather large and challenging topic. Achieving 'end-to-end' security requires a solution coping with many issues. The work presented in this thesis is not mean to give a complete answer to this challenge, or even a part of it. The methods and thoughts we use are a contribution to the vast knowledge related to the promising collaborative IS research.

Bibliography

[1] Dave King 0002, Trent Jaeger, Somesh Jha, and Sanjit A. Seshia. Effective blame for information-flow violations. In *SIGSOFT FSE*, pages 250–260, 2008.

[2] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4):706–734, 1993.

[3] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases: The Logical Level*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[4] Dakshi Agrawal, James Giles, Kang-Won Lee, and Jorge Lobo. Policy ratification. In *POLICY*, pages 223–232, 2005.

[5] Rakesh Agrawal, Peter J. Haas, and Jerry Kiernan. Watermarking relational data: framework, algorithms and analysis. *The VLDB Journal*, 12:157–169, August 2003.

[6] Rakesh Agrawal and Jerry Kiernan. Watermarking relational databases. In *VLDB*, pages 155–166, 2002.

[7] Berthold Agreiter, Muhammad Alam, Ruth Breu, Michael Hafner, Alexander Pretschner, Jean-Pierre Seifert, and Xinwen Zhang. A technical architecture for enforcing usage control requirements in service-oriented architectures. In *SWS*, pages 18–25, 2007.

[8] Bechara Al Bouna, Richard Chbeir, and Stefania Marrara. A multimedia access control language for virtual and ambient intelligence environments. In *SWS '07: Proceedings of the 2007 ACM workshop on Secure web services*, pages 111–120, New York, NY, USA, 2007. ACM.

[9] Mohammad A. Al-Kahtani and Ravi Sandhu. Induced role hierarchies with attribute-based RBAC. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 142–148, New York, NY, USA, 2003. ACM.

[10] Masoom Alam, Jean-Pierre Seifert, Qi Li, and Xinwen Zhang. Usage control platformization via trustworthy selinux. In *ASIACCS*, pages 245–248, 2008.

[11] Christopher. Alberts, Audrey. Dorofee, James. Stevens, and Carol. Woody. Introduction to the octave® approach. Technical report, 2003.

[12] Amazon. Amazon elastic computing cloud.

[13] ANSI. American national standard for information technology role based access control, 2004.

[14] Nadalin Anthony, Goodner Marc, Gudgin Martin, Barbir Abbie, and Granqvist Hans. OASIS WS-Trust 1.3, 2007.

[15] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise privacy authorization language (EPAL 1.2), 2003.

[16] Matthias Assel, Stefan Wesner, and Alexander Kipp. A security framework for dynamic collaborative working environments. *Identity in the Information Society*, 2:171–187, 2009. 10.1007/s12394-009-0027-1.

[17] B. Atkinson, G. Della-Libera, S. Hada, M. Hondo, P. Hallam-Baker, J. Klein, B. LaMacchia, P. Leach, J. Manferdelli, H. Maruyama, et al. Web services security (WS-Security), 2002.

[18] Franz Baader, Ian Horrocks, and Ulrike Sattler. Description logics as ontology languages for the semantic web. In *Festschrift in honor of Jörg Siekmann, Lecture Notes in Artificial Intelligence*, pages 228–248. Springer-Verlag, 2003.

[19] Philippe Balbiani, Jan Broersen, and Julien Brunel. Decision procedures for a deontic logic modeling temporal inheritance of obligations. *Electron. Notes Theor. Comput. Sci.*, 231:69–89, 2009.

[20] Sruthi Bandhakavi, Charles C. Zhang, and Marianne Winslett. Super-sticky and declassifiable release policies for flexible information dissemination control. In *WPES '06: Proceedings of the 5th ACM workshop on Privacy in electronic society*, pages 51–58, New York, NY, USA, 2006. ACM.

[21] Massimo Bartoletti, Pierpaolo Degano, Gian Luigi Ferrari, and Roberto Zunino. Model checking usage policies. pages 19–35, Berlin, Heidelberg, 2009. Springer-Verlag.

[22] Andreas Bauer and Jan Juerjens. Security protocols, properties, and their monitoring. In *SESS '08: Proceedings of the fourth international workshop on Software engineering for secure systems*, pages 33–40, New York, NY, USA, 2008. ACM.

[23] Eberhard Becker, Willms Buhse, Dirk Günnewig, and Niels Rump, editors. *Digital Rights Management - Technological, Economic, Legal and Political Aspects*, volume 2770 of *Lecture Notes in Computer Science*. Springer, 2003.

[24] Moritz Y. Becker and Peter Sewell. Cassandra: Flexible trust management, applied to electronic health records. In *CSFW '04: Proceedings of the 17th IEEE workshop on Computer Security Foundations*, page 139, Washington, DC, USA, 2004. IEEE Computer Society.

[25] Rajneeshkaur Bedi, Anita Thengade, and Wadhai M. Vijay. A new watermarking approach for non-numeric relational database. *International Journal of Computer Applications*, 13(7):37–40, January 2011. Published by Foundation of Computer Science.

[26] Meriam Ben-Ghorbel-Talbi, Frédéric Cuppens, Nora Cuppens-Boulahia, and Adel Bouhoula. Revocation schemes for delegation licences. In *ICICS*, pages 190–205, 2008.

[27] Meriam Ben-Ghorbel-Talbi, Frédéric Cuppens, Nora Cuppens-Boulahia, and Adel Bouhoula. A delegation model for extended rbac. *Int. J. Inf. Sec.*, 9(3):209–236, 2010.

[28] T. Berners-Lee. *Semantic web road map*, 1998.

[29] Agreiter Berthold, Muhammad Alam, Ruth Breu, Michael Hafner, Alexander Pretschner, Jean-Pierre Seifert, and Xinwen Zhang. A technical architecture for enforcing usage control requirements in service-oriented architectures. In *SWS '07: Proceedings of the 2007 ACM workshop on Secure web services*, pages 18–25, New York, NY, USA, 2007. ACM.

[30] Elisa Bertino, Carolyn Brodie, Seraphin B. Calo, Lorrie Faith Cranor, Clare-Marie Karat, John Karat, Ninghui Li, Dan Lin, Jorge Lobo, Qun Ni, Prathima Rao, and Xiping Wang. Analysis of privacy and security policies. *IBM Journal of Research and Development*, 53(2):3, 2009.

[31] Elisa Bertino, Wonjun Lee, Anna C. Squicciarini, and Bhavani Thuraisingham. End-to-end accountability in grid computing systems for coalition information sharing. In *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead*, CSIIRW '08, pages 29:1–29:3, New York, NY, USA, 2008. ACM.

[32] Clara Bertolissi and Maribel Fernández. A rewriting framework for the composition of access control policies. In *PPDP '08: Proceedings of the 10th international ACM SIGPLAN conference on Principles and practice of declarative programming*, pages 217–225, New York, NY, USA, 2008. ACM.

[33] Karthikeyan Bhargavan, Cédric Fournet, and Andrew D. Gordon. Verifying policy-based web services security. *ACM Trans. Program. Lang. Syst.*, 30(6), 2008.

[34] Frédérique Biennier. Security integration in inter-enterprise business process engineering. In Harinder S. Jagdev, J. C. Wortmann, and H. J. Pels, editors, *Collaborative systems for production management: IFIP TC5/WG5. 7 Eighth International Conference on Advances Production Management Systems, September 8-13, 2002, Eindhoven, the Netherlands*, pages 207–217. Springer Netherlands, 2003.

[35] Frédérique Biennier, Loubna Ali, and Anne Legait. Extended service integration: Towards manufacturing sla. In Jan Olhager and Fredrik Persson, editors, *Advances in Production Management Systems*, volume

246 of *IFIP International Federation for Information Processing*, pages 87–94. Springer Boston, 2007.

[36] Frédérique Biennier and Joel Favrel. Integration of a contract framework in bp models. *Virtual Enterprises and Collaborative Networks*, 149:97–104, 2004.

[37] Frédérique Biennier and Joel Favrel. Collaborative business and data privacy: Toward a cyber-control? *Computers in Industry*, 56(4):361–370, 2005.

[38] B. Boehm. A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes*, 11(4):14–24, 1986.

[39] Piero Bonatti and Pierangela Samarati. Regulating service access and information release on the web. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 134–143, New York, NY, USA, 2000. ACM.

[40] Piero A. Bonatti and Pierangela Samarati. A uniform framework for regulating service access and information release on the web. *J. Comput. Secur.*, 10(3):241–271, 2002.

[41] J. Bosak, T. McGrath, and G.K. Holman. Universal business language v2. 0, 2006.

[42] D. Box, F. Curbera, et al. Web services addressing (WS-Addressing), 2004.

[43] Krysia Broda, Keith Clark, Rob Miller 0002, and Alessandra Russo. Sage: A logical agent-based environment monitoring and control system. In *AmI*, pages 112–117, 2009.

[44] Glenn Bruns, Daniel S. Dantas, and Michael Huth. A simple and expressive semantic framework for policy composition in access control. In *FMSE*, pages 12–21, 2007.

[45] Glenn Bruns, Daniel S Dantas, and Michael Huth. A simple and expressive semantic framework for policy composition in access control. In *FMSE '07: Proceedings of the 2007 ACM workshop on Formal methods in security engineering*, pages 12–21, New York, NY, USA, 2007. ACM.

[46] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Trans. Comput. Syst.*, 8(1):18–36, 1990.

[47] Laurent Bussard, Gregory Neven, and Franz-Stefan Preiss. Downstream usage control. *Policies for Distributed Systems and Networks, IEEE International Workshop on*, 0:22–29, 2010.

[48] Barbara Carminati, Elena Ferrari, and Andrea Perego. Enforcing access control in Web-based social networks. *ACM Trans. Inf. Syst. Secur.*, 13(1), 2009.

[49] José Carmo and Olga Pacheco. Deontic and action logics for organized collective agency, modeled through institutionalized agents and roles. *Fundam. Inf.*, 48(2-3):129–163, 2001.

[50] Sodki Chaari, Youakim Badr, and Frédérique Biennier. Enhancing web service selection by qos-based ontology and ws-policy. In *SAC*, pages 2426–2431, 2008.

[51] David W. Chadwick and Stijn F. Lievens. Enforcing "sticky" security policies throughout a distributed application. In *MidSec '08: Proceedings of the 2008 workshop on Middleware security*, pages 1–6, New York, NY, USA, 2008. ACM.

[52] David W. Chadwick, Sassa Otenko, and Tuan Anh Nguyen. Adding support to XACML for multi-domain user to user dynamic delegation of authority. *Int. J. Inf. Secur.*, 8(2):137–152, 2009.

[53] Peter C. Chapin, Christian Skalka, and X. Sean Wang. Authorization in trust management: Features and foundations. *ACM Comput. Surv.*, 40(3):1–48, 2008.

[54] Scott Charney. Establishing end to end trust. 2008.

[55] Richard Chbeir and David Gross-Amblard. Multimedia and metadata watermarking driven by application constraints. In *MMM*, 2006.

[56] Wenzhi Chen, Zhipeng Zhang, Jianhua Yang, and Qinming He. vcerberus: A drtm system based on virtualization technology. *Wuhan University Journal of Natural Sciences*, 15:185–189, 2010. 10.1007/s11859-010-0301-y.

[57] R. Chinnici, J.J. Moreau, A. Ryman, and S. Weerawarana. Web services description language (WSDL) version 2.0 part 1: Core language, 2004.

[58] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1, 2001.

[59] Mandy Chung. Using JConsole to monitor applications. Technical report, Oracle, Sun Developer Network (SDN), 2004.

[60] Lorenzo Cirio, Isabel F. Cruz, and Roberto Tamassia. A role and attribute based access control system using semantic web technologies. In *OTM Workshops (2)*, pages 1256–1266, 2007.

[61] David D. Clark and David R. Wilson. A comparison of commercial and military computer security policies. *Security and Privacy, IEEE Symposium on*, 0:184, 1987.

[62] LLC Clark & Parsia. Pellet: Owl 2 reasoner for java.

[63] E. M. Clarke, E. A Emerson, and A. P Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications. Technical report, Austin, TX, USA, 1985.

[64] Cloud Hosting. Cloud computing and hybrid infrastructure from gogrid.

[65] CLUSIF. MEHARI: Information risk analysis and management methodology.

[66] Maurizio Colombo, Fabio Martinelli, Paolo Mori, and Aliaksandr Lazouski. On usage control for grid services. In *Proceedings of the 2009 International Joint Conference on Computational Sciences and Optimization - Volume 01*, pages 47–51, Washington, DC, USA, 2009. IEEE Computer Society.

[67] JBOSS community. jbossxacml.

[68] Camelia Constantin, David Gross-Amblard, and Meryem Guerrouani. Watermill: an optimized fingerprinting system for highly constrained data. In *Proceedings of the 7th workshop on Multimedia and security*, MM&Sec '05, pages 143–155, New York, NY, USA, 2005. ACM.

[69] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The platform for privacy preferences 1.0 (P3P1.0) specification, 2002.

[70] Robert Craven, Jorge Lobo, Emil Lupu, Alessandra Russo, and Morris Sloman. Security policy refinement using data integration: a position paper. In *Proceedings of the 2nd ACM workshop on Assurable and usable security configuration*, SafeConfig '09, pages 25–28, New York, NY, USA, 2009. ACM.

[71] Robert Craven, Jorge Lobo, Emil C. Lupu, Alessandra Russo, and Morris Sloman. Decomposition techniques for policy refinement. In *CNSM*, pages 72–79, 2010.

[72] Robert Craven, Jorge Lobo, Jiefei Ma, Alessandra Russo, Emil Lupu, and Arosha Bandara. Expressive policy analysis with enhanced system dynamicity. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 239–250, New York, NY, USA, 2009. ACM.

[73] Cloud Security Alliance (CSA). *Security Guidance for Critical Areas of Focus in Cloud Computing.*

[74] Frédéric Cuppens and Nora Cuppens-Boulahia. Modeling contextual security policies. *Int. J. Inf. Sec.*, 7(4):285–305, 2008.

[75] Frédéric Cuppens, Nora Cuppens-Boulahia, and Céline Coma. O2O: Virtual private organizations to manage security policy interoperability. In *ICISS*, pages 101–115, 2006.

[76] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *POLICY*, pages 18–38, 2001.

[77] Tim Berners-Lee Joan Feigenbaum James Hendler Gerald Jay Sussman Daniel J. Weitzner, Harold Abelson. Information accountability. Technical Report MIT-CSAIL-TR-2007-034, Computer Science and Artificial Intelligence Lab (CSAIL) of Massachusetts Institute of Technology (MIT), June 2007.

[78] Catteddu Daniele and Hogben Giles. Cloud Computing: Benefits, risks and recommendations for information security. Technical report, 2009.

[79] J. De Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, U. Keller, M. Kifer, B. K
"onig-Ries, J. Kopecky, R. Lara, et al. Web service modeling ontology (WSMO), 2005.

[80] J. De Bruijn, H. Lausen, A. Polleres, and D. Fensel. The web service modeling language wsml: An overview. *The Semantic Web: Research and Applications*, pages 590–604, 2006.

[81] Hervé Debar, Nizar Kheir, Nora Cuppens-Boulahia, and Frédéric Cuppens. Service dependencies in information systems security. In *MMM-ACNS*, pages 1–20, 2010.

[82] Li Ding, Dominic DiFranzo, Alvaro Graves, James Michaelis, Xian Li, Deborah L. McGuinness, and James A. Hendler. Twc data-gov corpus: incrementally generating linked government data from data.gov. In *WWW*, pages 1383–1386, 2010.

[83] Defense Nationale direction centrale de la securite des systemes d'information. The EBIOS® method, 2003.

[84] DOD. Virtual machine security technical implementation guide, 2005.

[85] Daniel J. Dougherty, Kathi Fisler, and Shriram Krishnamurthi. Obligations and their interaction with programs. *ESORICS 2007. LNCS*, 4734/2007:375–389, 2007.

[86] Siqing Du and James B. D. Joshi. Supporting authorization query and inter-domain role mapping in presence of hybrid role hierarchy. *Proceedings of the eleventh ACM symposium on Access control models and technologies - SACMAT '06*, pages 228–236, 2006.

[87] S.S.A.J.J. Dubray and MJ Martin. ebXML business process specification schema technical specification v2. 0.4, 2006.

[88] J. Duhl and S. Kevorkian. Understanding drm systems. *An IDC White Paper*, 2001.

[89] Petros Efstathopoulos and Eddie Kohler. Manageable fine-grained information flow. In *EuroSys*, pages 301–313, 2008.

[90] Robert J. Ellison and Andrew P. Moore. Architectural refinement for the design of survivable systems. Technical report, 2001.

[91] Jeff A. Estefan, K. Laskey, Francis G. McCabe, and Danny Thornton. Reference architecture for Service Oriented Architecture version 1.0, 2008.

[92] C. Evans, D. Chappell, D. Bunting, G. Tharakan, H. Shimamura, J. Durand, J. Mischkinsky, K. Nihei, K. Iwasa, M. Chapman, et al. Web services reliability (ws-reliability), ver. 1.0, 2003.

[93] D. Fensel and C. Bussler. The web service modeling framework WSMF. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[94] D. Ferraiolo, J. Cugini, and D.R. Kuhn. Role-based access control (RBAC): Features and motivations. In *Proceedings of 11th Annual Computer Security Application Conference*, pages 241–48, 1995.

[95] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[96] Jeanne Ferrante, Karl J. Ottenstein, and Joe D. Warren. The program dependence graph and its use in optimization. *ACM Trans. Program. Lang. Syst.*, 9(3):319–349, 1987.

[97] Rodolfo Ferrini and Elisa Bertino. Supporting RBAC with XACML+OWL. In *SACMAT*, pages 145–154, 2009.

[98] T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham. Rowlbac: representing role based access control in owl. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 73–82, New York, NY, USA, 2008. ACM.

[99] Flexiant. Flexiscale cloud comp and hosting.

[100] Achille Fokoue, Mudhakar Srivatsa, Pankaj Rohatgi, Peter Wrobel, and John Yesberg. A decision support system for secure information sharing. In *SACMAT*, pages 105–114, 2009.

[101] Simon N. Foley. The specification and implementation of commercial security requirements including dynamic segregation of duties. In *Proceedings of the 4th ACM conference on Computer and communications security*, CCS '97, pages 125–134, New York, NY, USA, 1997. ACM.

[102] Ian T. Foster. Globus toolkit version 4: Software for service-oriented systems. *J. Comput. Sci. Technol.*, 21(4):513–520, 2006.

[103] Eclipse Foundation. Eclipse test & performance tools platform project, 2011.

[104] Internet Education Foundation. *P3P Toolbox*.

[105] Cédric Fournet and Tamara Rezk. Cryptographically sound implementations for typed information-flow security. In *POPL*, pages 323–335, 2008.

[106] freeBSD. Freebsd handbook: File system access control lists.

[107] Borko Furht and Darko Kirovski. *Multimedia Watermarking Techniques and Applications (Internet and Communications Series)*. Auerbach Publications, Boston, MA, USA, 2006.

[108] Eimear Gallery and Chris J. Mitchell. Trusted computing: Security and applications. *Cryptologia*, 33(3):217–245, 2009.

[109] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, and Dan Boneh. Terra: a virtual machine-based platform for trusted computing. In *SOSP*, pages 193–206, 2003.

[110] G. Geethakumari, Atul Negi, and V N Sastry. A cross–domain role mapping and authorization framework for rbac in grid systems. *International Journal of Computer Science & Applications*, 6:1–12, 2009.

[111] Gabriela Gheorghe, Paolo Mori, Bruno Crispo, and Fabio Martinelli. Enforcing ucon policies on the enterprise service bus. In *OTM Conferences (2)*, pages 876–893, 2010.

[112] Gabriela Gheorghe, Stephan Neuhaus, and Bruno Crispo. xesb: An enterprise service bus for access and usage control policy enforcement. In *IFIPTM*, pages 63–78, 2010.

[113] Dorina Ghindici, Isabelle Simplot-Ryl, and Jean-Marc Talbot. A sound analysis for secure information flow using abstract memory graphs. In *FSEN*, pages 355–370, 2009.

[114] Christine Golbreich and Atsutoshi Imai. Combining SWRL rules and OWL ontologies with Protégé OWL Plugin , Jess , and Racer. *Most*, pages 2–5.

[115] S.L. Goldman, R.N. Nagel, and K. Preiss. *Agile competitors and virtual organizations: strategies for enriching the customer*. Van Nostrand Reinhold New York, 1995.

[116] Google code. Google app engine.

[117] GrammaTech. Dependence graphs and program slicing-codesurfer technology overview. Technical report, GrammaTech, Inc.

[118] David Gross-AMBLARD. Query-preserving watermarking of relational databases and xml documents. *ACM Trans. Database Syst.*, 36:3:1–3:24, March 2011.

[119] Information System Group. Hermit reasoner.

[120] Trusted Computing Group. Trusted Platform Module (TPM).

[121] Liang Gu, Yueqiang Cheng, Xuhua Ding, Robert H. Deng, Yao Guo, and Weizhong Shao. Remote attestation on function execution (work-in-progress). In *INTRUST*, pages 60–72, 2009.

[122] Liang Gu, Xuhua Ding, Robert Huijie Deng, Bing Xie, and Hong Mei. Remote attestation on program execution. In *STC*, pages 11–20, 2008.

[123] Bhavna Gupta, Harmeet Kaur, and Punam Bedi. A reputation based service provider selection system for farmers. *Journal of Information Assurance and Security*, 5:508–515, 2010.

[124] Volker Haarslev, Ralf Moller, Kay Hidde, and Michael Wessel. Racer-renamed abox and concept expression reasoner.

[125] H. Haas and A. Brown. *Web services glossary*, 2004.

[126] Hugo Haas. Designing the architecture for Web services. *W3C*, 2003.

[127] Chris Hanson, Tim Berners-Lee, Lalana Kagal, Gerald J. Sussman, and Daniel J. Weitzner. Data-purpose algebra: Modeling data usage policies. In *POLICY*, pages 173–177, 2007.

[128] William R. Harris, Nicholas Kidd, Sagar Chaki, Somesh Jha, and Thomas W. Reps. Verifying information flow control over unbounded processes. In *FM*, pages 773–789, 2009.

[129] Matús Harvan and Alexander Pretschner. State-based usage control enforcement with data flow tracking using system call interposition. In *NSS*, pages 373–380, 2009.

[130] Manuel Hilty, Alexander Pretschner, David A. Basin, Christian Schaefer, and Thomas Walter. A policy language for distributed usage control. *ESORICS2007. LNCS*, 4734/2007:531–546, 2008.

[131] Manuel Hilty, Alexander Pretschner, Christian Schaefer, and Thomas Walter. Enforcement for usage control- an overview of control mechanisms. *DoCoMo Euro-Labs Publication*, 2006.

[132] Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, Chris Wroe, Simon Jupp, Georgina Moulton, Robert Stevens, Nick Drummond, Simon Jupp, Georgina Moulton, and Robert Stevens. *A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.2*, 3 2009.

[133] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML, 2004.

[134] Hao Hu, Hao Li, and Dengguo Feng. L-UCON: Towards layered access control with UCON. In *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 02*, CSE '09, pages 823–829, Washington, DC, USA, 2009. IEEE Computer Society.

[135] Chen hua Ma, Guo-Dong Lu, and Jiong Qiu. An authorization model for collaborative access control. *Journal of Zhejiang University - Science C*, 11(9):699–717, 2010.

[136] R. Iannella. Digital rights management (DRM) architectures. *D-Lib Magazine*, 7(6), 2001.

[137] IBM. Using the JVM monitoring interface (JVMMI). Technical report, IBM, 2003.

[138] Shingo Inoue, Kyoko Makino, Ichiro Murase, Osamu Takizawa, Matsumoto Tsutomu, and Nakagawa Hiroshi. A proposal on information hiding methods using xml. In *Natural Language Processing Pacific Rim Symposium*.

[139] Keith Irwin, Ting Yu, and William H. Winsborough. On the modeling and analysis of obligations. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 134–143, New York, NY, USA, 2006. ACM.

[140] ISO/IEC. Iso 17799, 2002.

[141] ISO/IEC. INTERNATIONAL STANDARD ISO/IEC 27002: 2005(e):information technology security techniques code of practice for information security management, 2005.

[142] Helge Janicke, Antonio Cau, François Siewe, and Hussein Zedan. Concurrent enforcement of usage control policies. In *POLICY '08: Proceedings of the 2008 IEEE Workshop on Policies for Distributed Systems and Networks*, pages 111–118, Washington, DC, USA, 2008. IEEE Computer Society.

[143] Heiser Jay and Nicolett Mark. Assessing the security risks of Cloud Computing. Technical report, 6 2008.

[144] Kirch Joel. Virtual machine security guidelines version 1.0, 2007.

[145] Diana Jordan, John Evdemon, Alexandre Alves, Assaf Arkin, Sid Askary, Charlton Barreto, Ben Bloch, Francisco Curbera, Mark Ford, Yaron Goland, Alejandro Guzar, Neelakantan Kartha, Canyang Kevin Liu, Rania Khalaf, Dieter Knig, Mike Marin, Vinkesh Mehta, Satish Thatte, Danny van der Rijn, Prasad Yendluri, and Alex Yiu and. Web services Business Process Execution Language (WS-BPEL), April 2007.

[146] Diane Jordan, John Evdemon, et al. Web services business process execution language version 2.0, 2007.

[147] A Josang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, March 2006.

[148] L. Kagal, T. Finin, M. Paolucci, N. Srinivasan, K. Sycara, and G. Denker. Authorization and privacy for semantic web services. *Intelligent Systems, IEEE*, 19(4):50–56, 2005.

[149] Lalana Kagal and Hal Abelson. Access control is an inadequate framework for privacy protection, July 2010.

[150] Lalana Kagal, Tim Berners-Lee, Dan Connolly, and Daniel J. Weitzner. Using semantic web technologies for policy management on the web. In *AAAI*, 2006.

[151] Lalana Kagal, Chris Hanson, and Daniel J. Weitzner. Using dependency tracking to provide explanations for policy management. In *POLICY*, pages 54–61, 2008.

[152] Anas Abou El Kalam, Salem Benferhat, Alexandre Miège, Rania El Baida, Frédéric Cuppens, Claire Saurel, Philippe Balbiani, Yves Deswarte, and Gilles Trouessin. Organization based access contro. In *POLICY*, pages 120–, 2003.

[153] Clare-Marie Karat, John Karat, Carolyn Brodie, and Jinjuan Feng. Evaluating interfaces for privacy policy rule authoring. In *CHI*, pages 83–92, 2006.

[154] John Karat, Clare-Marie Karat, Elisa Bertino, Ninghui Li, Qun Ni, Carolyn Brodie, Jorge Lobo, Seraphin Calo, Lorrie Cranor, Ponnurangam Kumaraguru, and Robert Reeder. A policy famework for security and privacy management. *IBM System Journal*, special issue on harmonizing security and privacy.

[155] John Karat, Clare-Marie Karat, Elisa Bertino, Ninghui Li, Qun Ni, Carolyn Brodie, Jorge Lobo, Seraphin Calo, Lorrie Cranor, Ponnurangam Kumaraguru, and Robert Reeder. A policy framework for

security and privacy management. *IBM System Journal*, special issu on harmonizing security and privacy, Submitted.

[156] Basel Katt, Xinwen Zhang, Ruth Breu, Michael Hafner, and Jean-Pierre Seifert. A general obligation model and continuity: enhanced policy enforcement engine for usage control. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 123–132, New York, NY, USA, 2008. ACM.

[157] Basel Katt, Xinwen Zhang, and Michael Hafner. Towards a usage control policy specification with petri nets. In *OTM '09: Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems*, pages 905–912, Berlin, Heidelberg, 2009. Springer-Verlag.

[158] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. Web services choreography description language version 1.0, 2004.

[159] Nizar Kheir, Nora Cuppens-Boulahia, Frédéric Cuppens, and Hervé Debar. A service dependency model for cost-sensitive intrusion response. In *ESORICS*, pages 626–642, 2010.

[160] Nizar Kheir, Hervé Debar, Frédéric Cuppens, Nora Cuppens-Boulahia, and Jouni Viinikka. A service dependency modeling framework for policy-based response enforcement. In *DIMVA*, pages 176–195, 2009.

[161] Himanshu Khurana and Virgil D. Gligor. A model for access negotiations in dynamic coalitions. In *WETICE '04: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 205–210, Washington, DC, USA, 2004. IEEE Computer Society.

[162] Himanshu Khurana and Virgil D. Gligor. A model for access negotiations in dynamic coalitions. In *WETICE*, pages 205–210, 2004.

[163] A. Kim, J. Luo, and M. Kang. Security ontology for annotating resources. *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, pages 1483–1499, 2005.

[164] Puneet Kishor, Oshani Seneviratne, and Noah Giansiracusa. Policy aware geospatial data. In *ACM GIS09*. ACM Press, November 2009.

[165] Ilianna Kollia, Birte Glimm, and Ian Horrocks. Query answering over sroiq knowledge bases with sparql. In *Description Logics*, 2011.

[166] Vladimir Kolovski. Formal semantics of XACML v3.0. Technical report, University of Maryland, March 2008.

[167] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Gen. Comput.*, 4(1):67–95, 1986.

[168] Ram Krishnan, Jianwei Niu, Ravi Sandhu, and William H. Winsborough. Stale-safe security properties for group-based secure information sharing. In *FMSE '08: Proceedings of the 6th ACM workshop on Formal methods in security engineering*, pages 53–62, New York, NY, USA, 2008. ACM.

[169] Ram Krishnan, Ravi Sandhu, Jianwei Niu, and William H. Winsborough. A conceptual framework for group-centric secure information sharing. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 384–387, New York, NY, USA, 2009. ACM.

[170] Ram Krishnan, Ravi Sandhu, Jianwei Niu, and William H. Winsborough. Foundations for group-centric secure information sharing models. In *SACMAT '09: Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 115–124, New York, NY, USA, 2009. ACM.

[171] KVM.org. Kernel based virtual machine.

[172] Julien Lafaye, Jean Béguec, David Gross-Amblard, and Anne Ruas. Invisible graffiti on your buildings: blind and squaring-proof watermarking of geographical databases. In *Proceedings of the 10th international conference on Advances in spatial and temporal databases*, SSTD'07, pages 312–329, Berlin, Heidelberg, 2007. Springer-Verlag.

[173] Julien Lafaye and David Gross-Amblard. Xml streams watermarking. In *DBSec*, pages 74–88, 2006.

[174] Julien Lafaye, David Gross-Amblard, Camelia Constantin, and Meryem Guerrouani. Watermill: An optimized fingerprinting system for databases under constraints. *IEEE Trans. on Knowl. and Data Eng.*, 20:532–546, April 2008.

[175] Monica S. Lam, Michael C. Martin, V. Benjamin Livshits, and John Whaley. Securing web applications with static and dynamic information flow tracking. In *PEPM*, pages 3–12, 2008.

[176] Butler W Lampson. Protection. In *Proceedings of the 5th Princeton Conference on Information Sciences and Systems.*, page 437, 1971.

[177] H.L. Lee. The triple-A supply chain. *Harvard business review*, 82(10):102–113, 2004.

[178] Wonjun Lee, Anna Cinzia Squicciarini, and Elisa Bertino. Profile-based selection of accountability policies in grid computing systems. In *POLICY*, pages 145–148, 2011.

[179] Henry M. Levy. *Capability-Based Computer Systems.* Butterworth-Heinemann, Newton, MA, USA, 1984.

[180] Lei Li, Yan Wang, and Ee-Peng Lim. Trust-oriented composite service selection and discovery. In *ICSOC-ServiceWave '09: Proceedings of the 7th International Joint Conference on Service-Oriented Computing*, pages 50–67, Berlin, Heidelberg, 2009. Springer-Verlag.

[181] Ninghui Li, Benjamin Grosof, and Joan Feigenbaum. A practically implementable and tractable delegation logic. In *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, page 27, Washington, DC, USA, 2000. IEEE Computer Society.

[182] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Trans. Inf. Syst. Secur.*, 6(1):128–171, 2003.

[183] Ninghui Li, Mahesh V. Tripunitara, and Ziad Bizri. On mutually exclusive roles and separation-of-duty. *ACM Trans. Inf. Syst. Secur.*, 10, May 2007.

[184] Ninghui Li and Qihua Wang. Beyond separation of duty: An algebra for specifying high-level security policies. *J. ACM*, 55:12:1–12:46, August 2008.

[185] Ninghui Li, Qihua Wang, Wahbeh Qardaji, Elisa Bertino, Prathima Rao, Jorge Lobo, and Dan Lin. Access control policy combining: theory meets practice. In *SACMAT '09: Proceedings of the 14th ACM symposium on Access control models and technologies*, pages 135–144, New York, NY, USA, 2009. ACM.

[186] Ninghui Li, Qihua Wang, Wahbeh H. Qardaji, Elisa Bertino, Prathima Rao, Jorge Lobo, and Dan Lin. Access control policy combining: theory meets practice. In *SACMAT*, pages 135–144, 2009.

[187] Zude Li and Xiaojun Ye. Towards a dynamic multi-policy dissemination control model: (dmdcon). *SIGMOD Record*, 35(1):33–38, 2006.

[188] Dan Lin, Prathima Rao, Elisa Bertino, Ninghui Li, and Jorge Lobo. EXAM: a comprehensive environment for the analysis of access control policies. *Int. J. Inf. Sec.*, 9(4):253–273, 2010.

[189] Ban B. Linda, Cocchiara Richard, Lovejoy Kristin, Telford Ric, and Ernest Mark. The evolving role of IT managers and CIOs–findings from the 2010 IBM global IT risk study. Technical report, 2010.

[190] Rong Liu, Akhil Kumar, and Wil van der Aalst. A formal modeling approach for supply chain event management. *Decis. Support Syst.*, 43(3):761–778, 2007.

[191] Jaime Lloret, Miguel Garcia, Diana Bri, and Juan R. Diaz. A group-based content delivery network. In *UPGRADE '08: Proceedings of the third international workshop on Use of P2P, grid and agents for the development of content networks*, pages 37–44, New York, NY, USA, 2008. ACM.

[192] J. W. Lloyd. *Foundations of logic programming*. Springer-Verlag New York, Inc., New York, NY, USA, 1984.

[193] Floridi Luciano. *Information: A Very Short Introduction*. Oxford university press, Great Clarendon Street, Oxford ox2 6dp, 02 2010.

[194] Leonidas Lymberopoulos, Emil Lupu, and Morris Sloman. An adaptive policy based management framework for differentiated services networks. In *POLICY*, pages 147–158, 2002.

[195] Leonidas Lymberopoulos, Emil Lupu, and Morris Sloman. Ponder policy implementation and validation in a cim and differentiated services framework. In *NOMS (1)*, pages 31–44, 2004.

[196] C.M. MacKenzie, K. Laskey, F. McCabe, P.F. Brown, and R. Metz. Reference model for Service Oriented Architecture 1.0, 2006.

[197] Daniel W. Manchala. Trust metrics, models and protocols for electronic commerce transactions. In *ICDCS'1998. Proceedings. 18th International Conference on Distributed Computing Systems*, pages 312–321. IEEE, 1998.

[198] Srdjan Marinovic, Kevin P. Twidle, Naranker Dulay, and Morris Sloman. Teleo-reactive policies for managing human-centric pervasive services. In *CNSM*, pages 80–87, 2010.

[199] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, et al. OWL-S: Semantic markup for web services, 2004.

[200] Keen Martin, Acharya Amit, Bishop Susan, Hopkins Alan, Milinski Sven, Nott Chris, Robinson Rick, Adams Jonathan, and Verschueren Paul. Patterns: Implementing an soa using an enterprise service bus. Technical report, 7 2004.

[201] Fabio Martinelli, Paolo Mori, and Anna Vaccarelli. Towards continuous usage control on grid computational services. In *Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*, pages 82–, Washington, DC, USA, 2005. IEEE Computer Society.

[202] D.L. McGuinness, F. Van Harmelen, et al. OWL web ontology language overview, 2004.

[203] Stal Michael. Web Services: Beyond component-based computing. *Communication of the ACM*, (10):71–76, 2002.

[204] Microsoft. Windows azure.

[205] Mark Miller, Ka-Ping Yee, and Jonathan Shapiro. Capability myths demolished. Technical Report SRL2003-02, Systems Research Laboratory, Department of Computer Science, Johns Hopkins University. http://srl.cs.jhu.edu/pubs/SRL2003-02.pdf, 2003.

[206] Marie E G Moe, Bjarne E Helvik, and Svein J Knapskog. Comparison of the beta and the hidden markov models of trust in dynamic environments. *TM 2009, IFIP AICT 300*, pages 283–297, 2009.

[207] Jonathan D. Moffett and Morris S. Sloman. Policy conflict analysis in distributed system management. 1993.

[208] Marco Casassa Mont, Siani Pearson, and Pete Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In *DEXA '03: Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, page 377, Washington, DC, USA, 2003. IEEE Computer Society.

[209] Oliver Moser, Florian Rosenberg, and Schahram Dustdar. Non-intrusive monitoring and service adaptation for ws-bpel. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 815–824, New York, NY, USA, 2008. ACM.

[210] Antonio Muñoz, Antonio Maña, and Daniel Serrano. Protecting agents from malicious hosts using tpm. *IJCSA*, 6(5):30–58, 2009.

[211] K. Nance, M. Bishop, and B. Hay. Virtual machine introspection: Observation or interference? *Security & Privacy, IEEE*, (5):32–37, 10 2008.

[212] Daniele Nardi and Ronald J. Brachman. An introduction to description logics. In *Description Logic Handbook*, pages 1–40, 2003.

[213] Michael J. Nash and Keith R. Poland. Some conundrums concerning separation of duty. *Security and Privacy, IEEE Symposium on*, 0:201, 1990.

[214] Wilfred Ng and Ho Lam Lau. In *DASFAA*, pages 68–80, 2005.

[215] Qun Ni and Elisa Bertino. xfacl: an extensible functional language for access control. In *SACMAT*, pages 61–72, 2011.

[216] Qun Ni, Elisa Bertino, and Jorge Lobo. An obligation model bridging access control policies and privacy policies. In *SACMAT*, pages 133–142, 2008.

[217] Qun Ni, Elisa Bertino, and Jorge Lobo. D-algebra for composing access control policy decisions. In *ASIACCS*, pages 298–309, 2009.

[218] Qun Ni, Elisa Bertino, and Jorge Lobo. D-algebra for composing access control policy decisions. In *ASIACCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 298–309, New York, NY, USA, 2009. ACM.

[219] Qun Ni, Shouhuai Xu, Elisa Bertino, Ravi S. Sandhu, and Weili Han. An access control language for a general provenance model. In *Secure Data Management*, pages 68–88, 2009.

[220] NIST. Definition of Cloud Computing v15.

[221] Donald Norman. *The design of everyday things*. Basic Books, New York, 1988.

[222] OASIS. OASIS Identity in the Cloud TC.

[223] OASIS. Universal Description, Discovery, and Integration (UDDI)3.0.2, 2004.

[224] OASIS. eXtensible Access Control Markup Language (XACML) version 2.0, 2005.

[225] OASIS. Quality model for web services, 2005.

[226] OASIS. Security Assertion Markup Language v2.0, 2005.

[227] Initiative ODRL. *Open Digital Rights Language (ODRL) Version 2 Requirements*.

[228] University of Manchester. Fact++.

[229] Anthony M. O'Hagan, Shazia Sadiq, and Wasim Sadiq. Evie - a developer toolkit for encoding service interaction patterns. *Information Systems Frontiers*, 11(3):211–225, 2009.

[230] Cloud Standards ORG. Cloud Standards Coordination.

[231] M. Paolucci and K. Sycara. Autonomous semantic web services. *IEEE Internet Computing*, 7(5):34–41, 2003.

[232] Panagiotis Papadimitriou and Hector Garcia-Molina. A model for data leakage detection. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 1307–1310, Washington, DC, USA, 2009. IEEE Computer Society.

[233] M. Papazoglou and D. Georgakopoulos. Introduction to a special issue on service oriented computing. *Communication of the ACM*, (10):25–28, 2003.

[234] Mike P. Papazoglou and Willem-Jan Ven Den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, March 2007.

[235] M.P. Papazoglou. Extending the service oriented architecture. *Journal of Business Integration*, (10):18–21, 2 2005.

[236] M.P. Papazoglou and J. Dubray. A survey of web service technologies. Technical report, University of Trento, 2004.

[237] Jaehong Park and Ravi Sandhu. Towards usage control models: beyond traditional access control. In *SACMAT '02: Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 57–64, New York, NY, USA, 2002. ACM.

[238] Jaehong Park and Ravi Sandhu. The UCON_ABC usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.

[239] Bryan Parno. Bootstrapping trust in a "trusted" platform. In *HotSec*, 2008.

[240] Bryan Parno, Jonathan M. McCune, and Adrian Perrig. Bootstrapping trust in commodity computers. In *IEEE Symposium on Security and Privacy*, pages 414–429, 2010.

[241] Marcus Peinado, Paul England, and Yuqun Chen. An overview of ngscb. *Trusted Computing*, IEE Perofessional Applications of Computing Series 6:115–141, 2005.

[242] H. Perritt. Permission headers and contract law. In *Proceedings of the Workshop on Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment*, 1994.

[243] PEtALS. Petals esb, distributed service platform.

[244] PEtALS. Petals master, soa governance solution.

[245] Fingar Peter and Bellini Joe. *Fingar P, Bellini J (2004) The real-time enterprise: competing on time using the revolutionary business SEx machine. Meghan-Kiffer Press, Tampa*. Meghan-Kiffer Press, 2004.

[246] Stere Preda, Frédéric Cuppens, Nora Cuppens-Boulahia, Joaquín García-Alfaro, Laurent Toutain, and Yehia Elrakaiby. Semantic context aware security policy deployment. In *ASIACCS*, pages 251–261, 2009.

[247] Alexander Pretschner, Manuel Hilty, and David Basin. Distributed usage control. *Commun. ACM*, 49(9):39–44, 2006.

[248] Alexander Pretschner, Manuel Hilty, Florian Schütz, Christian Schaefer, and Thomas Walter. Usage control enforcement: Present and future. *IEEE Security and Privacy*, 6(4):44–53, 2008.

[249] Alexander Pretschner, Judith Rüesch, Christian Schaefer, and Thomas Walter. Formal analyses of usage control policies. In *ARES*, pages 98–105, 2009.

[250] Yu Qin, Dengguo Feng, and Chunyong Liu. Tpm context manager and dynamic configuration management for trusted virtualization platform. *Wuhan University Journal of Natural Sciences*, 13:539–546, 2008. 10.1007/s11859-008-0506-5.

[251] Rackspace Hosting. Managed hosting and cloud hosting by rackspace hosting.

[252] Prathima Rao, Dan Lin, Elisa Bertino, Ninghui Li, and Jorge Lobo. Fine-grained integration of access control policies. *Computers & Security*, 30(2-3):91–107, 2011.

[253] Robert W. Reeder, Lujo Bauer, Lorrie Faith Cranor, Michael K. Reiter, Kelli Bacon, Keisha How, and Heather Strong. Expandable grids for visualizing and authoring computer security policies. In *CHI*, pages 1473–1482, 2008.

[254] Robert W. Reeder, Lujo Bauer, Lorrie Faith Cranor, Michael K. Reiter, and Kami Vaniea. More than skin deep: measuring effects of the underlying model on access-control system usability. In *CHI*, pages 2065–2074, 2011.

[255] B. Rosenblatt. Integrating DRM with peer-to-peer networks: Enabling the future of online content business models. *Whitepaper GiantSteps–Media Technologies Strategies*, 2003.

[256] Bill. Rosenblatt, Bill. Trippe, and Stephen. Mooney. Digital rights management: Business and technology. *M&T Books,*, 2003.

[257] Indrajit Roy, Donald E. Porter, Michael D. Bond, Kathryn S. McKinley, and Emmett Witchel. Laminar: practical fine-grained decentralized information flow control. In *PLDI*, pages 63–74, 2009.

[258] Giovanni Russello and Naranker Dulay. xDUCON: Coordinating usage control policies in distributed domains. In *Proceedings of the 2009 Third International Conference on Network and System Security*, NSS '09, pages 246–253, Washington, DC, USA, 2009. IEEE Computer Society.

[259] Giovanni Russello and Naranker Dulay. xDUCON: cross domain usage control through shared data spaces. In *Proceedings of the 10th IEEE international conference on Policies for distributed systems and networks*, POLICY'09, pages 178–181, Piscataway, NJ, USA, 2009. IEEE Press.

[260] Reiner Sailer, Trent Jaeger, Xiaolan Zhang, and Leendert van Doorn. Attestation-based policy enforcement for remote access. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 308–317, New York, NY, USA, 2004. ACM.

[261] Salesforce. Force.com.

[262] Ravi Sandhu. Transaction control expressions for separation of duties. In *Proc. of the Fourth Computer Security Applications Conference*, pages 282–286, 1988.

[263] Ravi. Sandhu, David Ferraiolo, and Richard Kuhn. The NIST model for role-based access control: Towards a uni ed standard. In *Proceedings of 5th ACM Workshop on Role-Based Access Control (Berlin, Germany.* ACM, 2000.

[264] Ravi S. Sandhu and Xinwen Zhang. Peer-to-peer access control architecture using trusted computing technology. In *SACMAT*, pages 147–158, 2005.

[265] Thierry Sans, Frédéric Cuppens, and Nora Cuppens-Boulahia. Form : A federated rights expression model for open drm frameworks. In *ASIAN*, pages 45–59, 2006.

[266] SAP. SAP Business ByDesign.

[267] Christian Schaefer. Usage control reference monitor architecture. *Security, Privacy and Trust in Pervasive and Ubiquitous Computing, International Workshop on*, 0:13–18, 2007.

[268] Oshani Seneviratne and Lalana Kagal. Addressing data reuse issues at the protocol level. In *POLICY*, pages 141–144, 2011.

[269] Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. Watermarking relational databases using optimization based techniques. Technical Report 41, Center for Education and Research in Information Assurance and Security (CERIAS), Purdue University, 2006.

[270] ChangXiang Shen, HuanGuo Zhang, HuaiMin Wang, Ji Wang, Bo Zhao, Fei Yan, FaJiang Yu, LiQiang Zhang, and MingDi Xu. Research on trusted computing and its development. *SCIENCE CHINA Information Sciences*, 53:405–433, 2010. 10.1007/s11432-010-0069-x.

[271] Zhidong Shen and Xiaoping Wu. A trusted computing technology enabled mobile agent system. In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume*

03, CSSE '08, pages 567–570, Washington, DC, USA, 2008. IEEE Computer Society.

[272] Avraham Shinnar, Marco Pistoia, and Anindya Banerjee. A language for information flow: dynamic tracking in multiple interdependent dimensions. In *PLAS*, pages 125–131, 2009.

[273] Jatinder Singh, David M. Eyers, and Jean Bacon. Controlling historical information dissemination in publish/subscribe. In *Middleware Security*, pages 34–39, 2008.

[274] Jatinder Singh, Luis Vargas, Jean Bacon, and Ken Moody. Policy-based information sharing in publish/subscribe middleware. In *POLICY*, pages 137–144, 2008.

[275] Radu Sion. Database watermarking. *Lecture Notes available online*.

[276] Radu Sion, Mikhail Atallah, and Sunil Prabhakar. Rights protection for relational data. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003.

[277] Radu Sion, Mikhail J. Atallah, and Sunil Prabhakar. Resilient rights protection for sensor streams. In *VLDB*, pages 732–743, 2004.

[278] Badri Siraman and Rakesh Radhakrishnan. Event driven architecture augmenting Service Oriented Architecture. Technical report, 1 2005.

[279] Layth Sliman, Frédérique Biennier, and Youakim Badr. A security policy framework for context-aware and user preferences in e-services. *J. Syst. Archit.*, 55(4):275–288, 2009.

[280] M. Sloman. Policy driven management for distributed systems. *Journal of network and Systems Management*, 2(4):333–360, 1994.

[281] Morris Sloman and Emil C. Lupu. Engineering policy-based ubiquitous systems. *Comput. J.*, 53(7):1113–1127, 2010.

[282] D. Smith. Web services enable service oriented and eventdriven architectures. *Journal of Business Integration*, pages 12–13, 5 2004.

[283] Anna Cinzia Squicciarini, Wonjun Lee, Elisa Bertino, and Carol X. Song. A policy-based accountability tool for grid computing systems. In *APSCC*, pages 95–100, 2008.

[284] Mudhakar Srivatsa, Shane Balfe, Kenneth G. Paterson, and Pankaj Rohatgi. Trust management for secure information flows. In *ACM Conference on Computer and Communications Security*, pages 175–188, 2008.

[285] Mudhakar Srivatsa, Pankaj Rohatgi, Shane Balfe, and Steffen Reidt. Securing information flows: A metadata framework. In *MASS*, pages 748–753, 2008.

[286] Yuzhong Sun, Haifeng Fang, Ying Song, Lei Du, Kai Zhang, Hongyong Zang, Yaqiong Li, Yajun Yang, Ran Ao, and Yongbing Huang. Trainbow: a new trusted virtual machine based platform. *Frontiers of Computer Science in China*, 4(1):47–64, 2010.

[287] SUN microsytem. The SUN XACML implementation.

[288] CA technologies. wrap it up! why application wrapping can help msps build margin.

[289] Rafael Teigao, Carlos Maziero, and Altair Olivo Santin. Applying a usage control model in an operating system kernel. *J. Network and Computer Applications*, 34(4):1342–1352, 2011.

[290] Oliver Thomas and Jan vom Brocke. A value-driven approach to the design of service-oriented information systemsmaking use of conceptual models. *Information Systems and e-Business Management*, 8(1):67–97, February 2009.

[291] Janice Y. Tsai, Serge Egelman, Lorrie Faith Cranor, and Alessandro Acquisti. The effect of online privacy information on purchasing behavior: An experimental study. *Information Systems Research*, 22(2):254–268, 2011.

[292] A. Tsalgatidou and T. Pilioura. An overview of standards and related technology in Web services. *Distributed and Parallel Databases*, 12(2):135–162, 2002.

[293] Kevin P. Twidle, Naranker Dulay, Emil Lupu, and Morris Sloman. Ponder2: A policy system for autonomous pervasive environments. In *ICAS*, pages 330–335, 2009.

[294] Luis M. Vaquero, Luis Rodero-Merino, and Daniel Morán. Locking the sky: a survey on IaaS cloud security. *Computing*, 9 2010.

[295] S.L. Vargo and R.F. Lusch. Evolving to a new dominant logic for marketing. *Journal of marketing*, 68(1):1–17, 2004.

[296] Inc VMware. Vmware.

[297] W3C. Mathematical Markup Language (MathML) version 2.0 (second edition).

[298] W3C. XML Path language (XPath) version 1.0.

[299] W3C. XQuery 1.0: An XML query language.

[300] W3C. XQuery 1.0 and XPath 2.0 functions and operators.

[301] Hua Wang, Jinli Cao, and Yanchun Zhang. Delegating revocations and authorizations in collaborative business environments. *Information Systems Frontiers*, 11(3):293–305, 2009.

[302] Junbo Wang, Zixue Cheng, Lei Jing, Kaoru Ota, and Mizuo Kansen. A smart-gate based composition method to provide services by solving conflict using dynamic user priority and compromise policy. *International Journal of Innovative Computing, Information and Control (IJICIC)*, page 2987.

[303] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. A logic-based framework for attribute based access control. In *FMSE '04: Proceedings of the 2004 ACM workshop on Formal methods in security engineering*, pages 45–55, New York, NY, USA, 2004. ACM.

[304] Qihua Wang, Hongxia Jin, and Ninghui Li. Usable access control in collaborative environments: Authorization based on people-tagging. In *ESORICS*, pages 268–284, 2009.

[305] X. Wang, G. Lao, T. DeMartini, H. Reddy, M. Nguyen, and E. Valenzuela. XrML–extensible rights markup language. In *Proceedings of the 2002 ACM workshop on XML security*, pages 71–79. ACM, 2002.

[306] Yulin Wang, Yang Shen, and Jian Pan. Usage control based on windows kernel hook. In *Proceedings of the 2009 International Conference on Information and Multimedia Technology*, ICIMT '09, pages 264–267, Washington, DC, USA, 2009. IEEE Computer Society.

[307] Daniel J. Weitzner, Harold Abelson, Tim Berners-Lee, Joan Feigenbaum, James A. Hendler, and Gerald J. Sussman. Information accountability. *Commun. ACM*, 51(6):82–87, 2008.

[308] Ingo Westphal, Klaus-Dieter Thoben, and Marcus Seifert. Managing collaboration performance to govern virtual organizations. *Journal of Intelligent Manufacturing*, 21(3):311–320, November 2008.

[309] James P. Womack and Daniel T. Jones. *Lean thinking, second edition.* Simon & Schuster New York, 2003.

[310] Zhengping Wu. A comparative study of trust management for federation: features and measurements. *Journal of Information Assurance and Security*, 6:029–038, 2011.

[311] Xen.org. The xen hypervisor.

[312] Daoxi Xiu and Zhaoyu Liu. A formal definition for trust in distributed systems. In *ISC*, pages 482–489, 2005.

[313] Wei Xu, Ling Huang, Armando Fox, David A. Patterson, and Michael I. Jordan. Mining console logs for large-scale system problem detection. In *SysML*, 2008.

[314] Wei Xu, Ling Huang, Armando Fox, David A. Patterson, and Michael I. Jordan. Online system problem detection by mining patterns of console logs. In *ICDM*, pages 588–597, 2009.

[315] Wei Xu, Ling Huang, Armando Fox, David A. Patterson, and Michael I. Jordan. Detecting large-scale system problems by mining console logs. In *ICML*, pages 37–46, 2010.

[316] Raj Yavatkar, Dimitrios Pendarakis, and Roch Guerin. Rfc2753: A framework for policy-based admission control. Network Working Group Request for Comments: 2753, Internet Engineering Task Force, January 2003.

[317] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed. Deploying and managing web services: issues, solutions, and directions. *The International Journal on Very Large Data Bases*, 17(3):537–572, 2008.

[318] Ting Yu, Xiaosong Ma, and Marianne Winslett. PRUNES: an efficient and complete strategy for automated trust negotiation over the internet. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 210–219, New York, NY, USA, 2000. ACM.

[319] Ting Yu, Marianne Winslett, and Kent E. Seamons. Interoperable strategies in automated trust negotiation. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 146–155, New York, NY, USA, 2001. ACM.

[320] Ting Yu, Marianne Winslett, and Kent E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Trans. Inf. Syst. Secur.*, 6(1):1–42, 2003.

[321] Yu Yu, Jussipekka Leiwo, and Benjamin Premkumar. How to privately utilize untrustworthy computing power. *Journal of Information Assurance and Security*, 1:149–158, 2006.

[322] Liangzhao Zeng, Boualem Benatallah, Marlon Dumas, Jayant Kalagnanam, and Quan Z. Sheng. Quality driven web services composition. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 411–421, New York, NY, USA, 2003. ACM.

[323] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, April 2010.

[324] Xinwen Zhang, Yingjiu Li, and Divya Nalla. An attribute-based access matrix model. In *SAC*, pages 359–363, 2005.

[325] Xinwen Zhang, Masayuki Nakae, Michael J. Covington, and Ravi Sandhu. A usage-based authorization framework for collaborative computing systems. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 180–189, New York, NY, USA, 2006. ACM.

[326] Xinwen Zhang, Masayuki Nakae, Michael J. Covington, and Ravi Sandhu. Toward a usage-based security framework for collaborative computing systems. *ACM Trans. Inf. Syst. Secur.*, 11:3:1–3:36, February 2008.

[327] Xinwen Zhang, Francesco Parisi-Presicce, Ravi Sandhu, and Jaehong Park. Formal model and policy specification of usage control. *ACM Trans. Inf. Syst. Secur.*, 8(4):351–387, 2005.

[328] Xinwen Zhang, Jaehong Park, Francesco Parisi-Presicce, and Ravi Sandhu. A logical specification for usage control. In *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 1–10, New York, NY, USA, 2004. ACM.

[329] Xinwen Zhang, Jean-Pierre Seifert, and Ravi Sandhu. Security enforcement model for distributed usage control. In *SUTC '08: Proceedings of the 2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*, pages 10–18, Washington, DC, USA, 2008. IEEE Computer Society.

[330] Hang Zhao, Jorge Lobo, Arnab Roy 0003, and Steven M. Bellovin. Policy refinement of network services for manets. In *Integrated Network Management*, pages 113–120, 2011.

[331] Jianjun Zhao and Martin Rinard. System dependence graph construction for aspect-oriented programs, 2003.

[332] Lantian Zheng and Andrew C. Myers. Dynamic security labels and static information flow control. *Int. J. Inf. Sec.*, 6(2-3):67–84, 2007.

[333] Xuan Zhou, HweeHwa Pang, Kian-Lee Tan, and Dhruv Mangla. Wmxml: A system for watermarking xml data. In *VLDB*, pages 1318–1321, 2005.

[334] Deqing Zou, Weide Zheng, Jinjiu Long, Hai Jin, and Xueguang Chen. Constructing trusted virtual execution environment in p2p grids. *Future Generation Comp. Syst.*, 26(5):769–775, 2010.

N° d'ordre 2012ISAL0027 Année 2012

Thèse

Application de gestion des droits numériques au système d'information d'entreprise

Présentée devant
L'institut national des sciences appliquées de Lyon
Pour obtenir
Le grade de docteur

Formation doctorale : Informatique
École doctorale : Informatique et Mathématique de Lyon
Par
Ziyi SU
Sous la Direction de :
Frédérique Biennier (Professeur à INSA de Lyon)

Soutenue le **22 mars 2012** devant la Commission d'examen

Jury MM.

Rapporteur**Prof. Ernesto DAMIANI**
Rapporteur**Dr. Khalid BENALI**
Dr. Agnès FRONT
Prof. Chirine GHEDIRA-GUEGAN
Prof. Bruno VALLESPIR
Prof. Frédérique BIENNIER

Laboratoire de recherche : Laboratoire d'InfoRmatique en Image et Systèmes d'information (**LIRIS**)

Context

Figures

Introduction

Pour faire face aux enjeux d'une économie mondialisée et aux fluctuations du marché, les entreprises (et principalement les PMEs) doivent développer de nouvelles stratégies de collaboration et d'organisation en réseau pour réduire les couts et se recentrer sur leur cœur de métier. Ce type d'organisation collaborative utilise largement les technologies de l'information et de la communication qui permettent d'augmenter l'agilité des l'entreprises et partager efficacement des données et connaissances et donc interconnecter leur systèmes. Cette stratégie de collaboration pose le problème d'une gestion de la sécurité « de bout en bout » pendant tout le cycle de vie des informations et processus partagés. Or les approches traditionnelles de sécurité ne proposent qu'une protection « statique » et « instantanée » ce qui ne permet pas de répondre aux contraintes de gestion des usage et d'adaptation dynamique requises par les organisations collaboratives (figure 1).

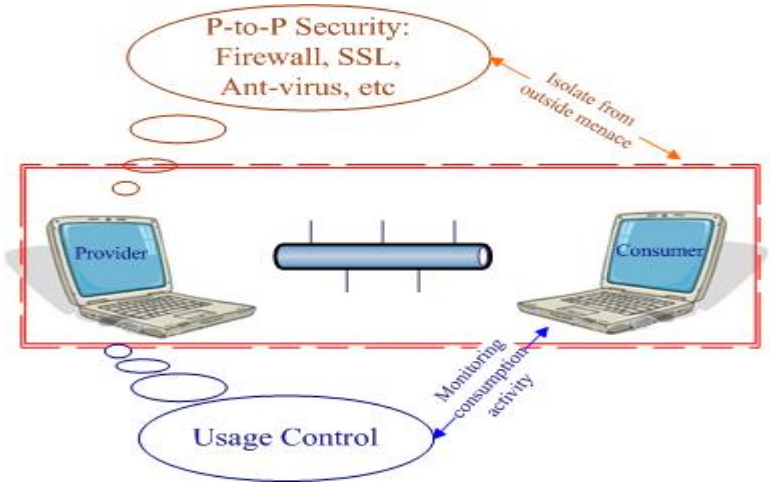


Figure 1. Sécurité de bout-en-bout

Le développement des stratégies collaborative pose le problème ,des risques liés à l'utilisation abusive de biens partagés par les partenaires et donc d'atteinte à la propriété intellectuelle. Ceci conduit à la montée des exigences de sécurité. Les

enquêtes menées par IBM [16], Gartner [12], le MIT [14] et l'agence européenne de la sécurité de l'information (ENISA) [5] montrent que les risques dus à la « manipulation des données sensibles par un tiers », la « perte de la gouvernance » et les menaces de violation de données ou de perte sont des obstacles majeurs au développement de ces stratégies de collaboration et d'ouverture. On note que ces mêmes risques et menaces concernent aussi la sphère des systèmes d'information avec les paradigmes SOA et Cloud.

Pour remédier à ces risques, il convient d'apporter la sécurité « de bout en bout » pour assurer la protection des biens des entreprises tout au long de leur cycle de vie (et pas seulement pendant la phase d'échange). Cet objectif implique la protection des biens vis-à-vis des utilisations non autorisées et des menaces extérieures.

Atteindre cet objectif de protection suppose que :

- Les participants puissent être capables d'exprimer leurs besoins de protection grâce à un modèle de politique.
- Les attributs de sécurité de chaque partenaire soient rendus visibles pour les autres,
- Un mécanisme d'appui à la négociation concernant la sécurité basée sur les attributs et les politiques soit défini.
- L'application des politiques puisse intégrer des modules de surveillance pour les systèmes des participants.

Notre approche est basée sur les médologies de DRM pour mettre en place une protection de bout en bout, c'est-à-dire pouvoir protéger les biens pendant tout leur cycle de vie dans un contexte de collaboration.

État de l'art

SOA: un nouveau paradigme de fédération d'enterprise

Les Architectures Orientées Services (SOA) offrent une nouvelle façon de modéliser et mettre en œuvre les systèmes d'information. Un service est un mécanisme permettant d'accéder à une ou plusieurs fonctionnalités, en utilisant une interface prescrits et les contraintes d'exécution / politiques spécifiées dans la description du service [17]. Compte tenu de l'introduction de nombreux standards, la stratégie SOA permet d'ouvrir le système d'information et lui offre plus d'agilité ce qui le rend adapté au contexte de la collaboration.

Les éléments fondamentaux du modèle SOA sont (voir la figure 2) :

- Le fournisseur de service (« service provider ») est la partie qui fournit des ressources (les services) pour répondre à des besoins spécifiques .
- Le consommateur de service (« service consumer ») est le participant qui a un besoin qui peut être rempli par un service.

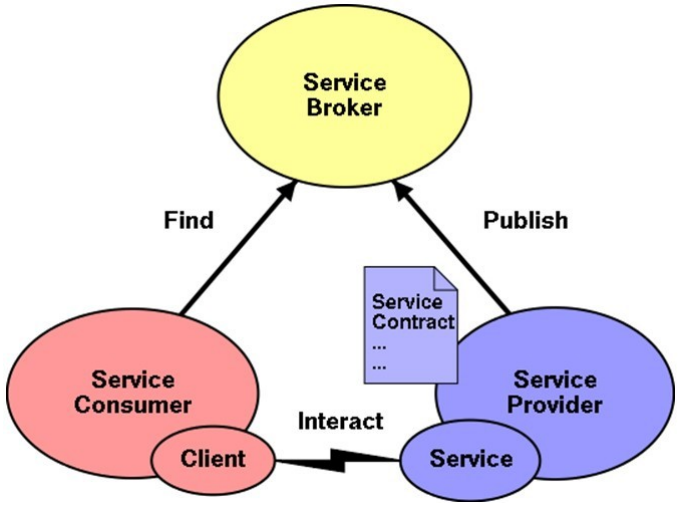


Figure 2. Eléments fondamentaux d'une SOA [10]

- Le courtier de service (« service broker ») permet aux fournisseurs de publier leurs services et aux consommateurs de services de rechercher les services dont ils ont besoin. Le courtier est semblable à un service de « pages jaunes ».

Le recours à une stratégie « orientées services » a plusieurs impacts sur l'organisation de l'infrastructure du système d'information des entreprises :

- La SOA offre une nouvelle façon de modéliser les fédérations des entreprises.

- La SOA est évolutive et apporte de l'agilité pour la modélisation des processus d'affaires puisqu'on peut, en recherchant les services adaptés et en les composants, construire des processus ad hoc.

- La SOA permet de définir des politiques pour les contrats portant sur la sécurité et la qualité de service (QoS).

Mise en œuvre d'une SOA

Un « Service Web » est défini par le W3C comme « un système logiciel conçu pour permettre l'interopérabilité de machine-à-machine sur un réseau » [9]. Les étapes du cycle de vie d'un Service Web comprennent la création, la description, la publication, la découverte, l'invocation et l'abandon (voir figure 3). La couche « valeur ajoutée » couvre les activités qui apportent de meilleures performances à l'environnement Web Service (voir figure 4).

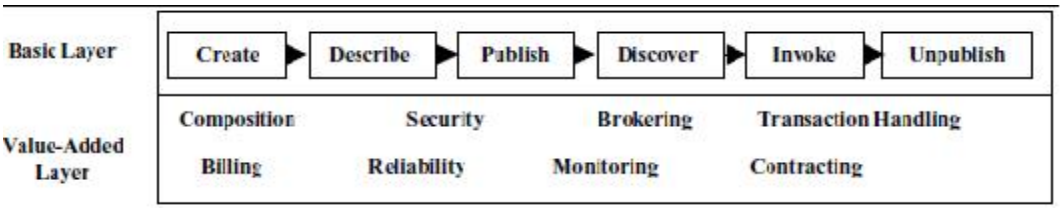


Figure 3. Les activités du cycle de vie de Service Web [27]

Dans l'organisation des piles liées aux services web, on trouve :

- Les couches de base qui sont consacrées à la communication via le réseau, en utilisant principalement SOAP, WS-Routing, WS-Addressing [2], WS-reliability [8].

- Les couches de haut niveau permettent de gérer la description des Services Web, en utilisant WSDL [3] [4], xCBL, cXML, UBL [1] et ebXML [6], etc. Ces couches permettent aussi la publication de Service Web dans des annuaires (en utilisant UDDI (Universal Description Discovery and Integration par exemple) [19].

- La coordination, l'orchestration et la chorégraphie permettent de Service Web sont des questions concernant la gestion des Web Services de coopération. Les standard plus importants pour mettre en œuvre ces tâches sont WS-BPSS, WS-BPEL [13] et WS-CDL [15].

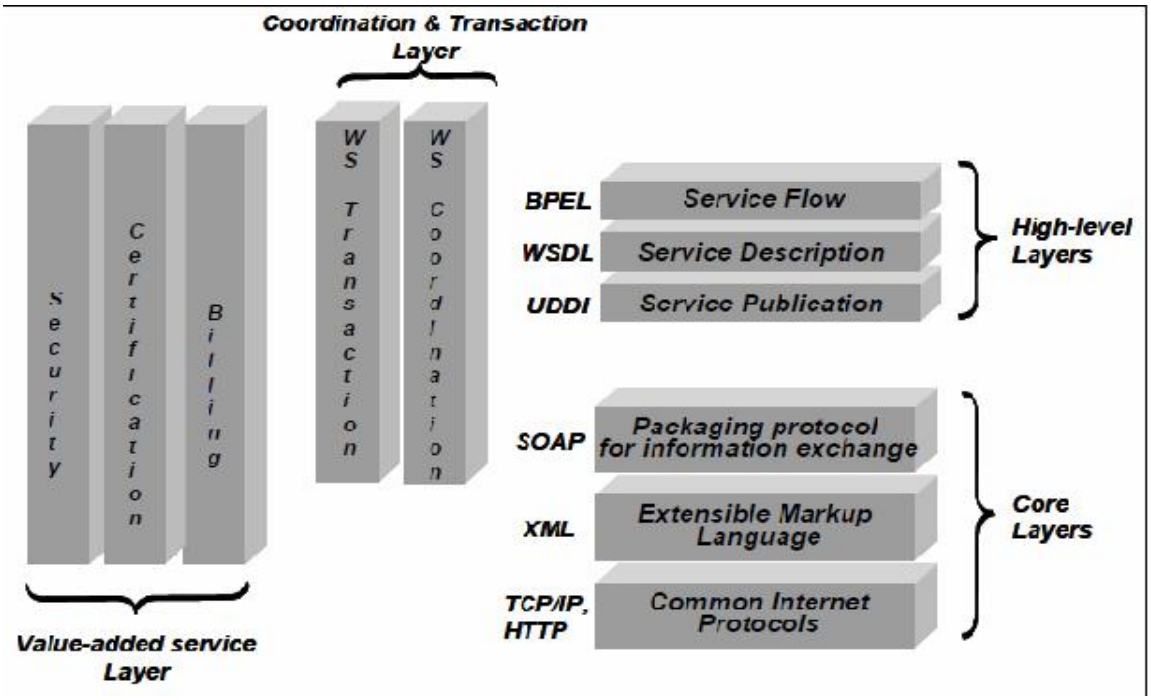


Figure 4. La pile de technologie web services [23]

Mettre en œuvre une stratégie SOA suppose la réorganisation des systèmes d'information pour intégrer ces nouvelles technologies tout en garantissant la sécurité des données et des services d'une part et d'intégrer des middleware permettant de mettre en œuvre efficacement ces technologies d'autres part.

L'approche « xSOA » ou « SOA étendu » [20] [22] peut permettre de répondre à ces exigences. En effet, la « xSOA » est une architecture multi-niveaux qui intègre une séparation multidimensionnelle des préoccupations fondées sur la nécessité de séparer des capacités de service « de baseé » et de permettre grâce à la gestion de leurs fonctionnalités et propriétés non fonctionnelles de construire simultanément des services plus élaborés et de permettre leur management (voir figure 5).

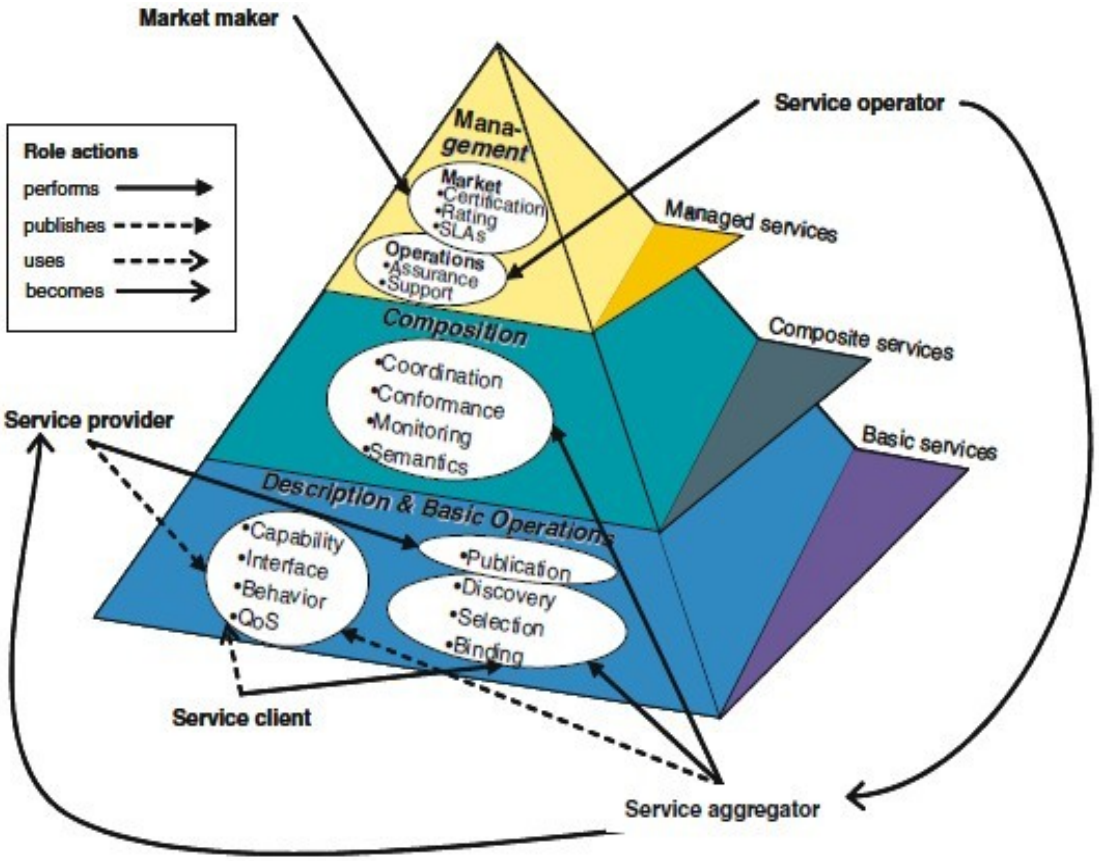


Figure 5. SOA étendu [21]

Pour ce qui concerne l'intégration d'un support aux déploiement de solutions orientées services on peut mettre en place un middleware « Enterprise Service Bus » (ESB) offrant des fonctionalités destinées à supporter l'échange de message, la composition, le routage… en utilisant des « connecteurs » standardisés. L'intergiciel PEtALS est une solution open source offrant ces fonctionnalités. En outre, cette offre intègre d'autres composants (gestion d'annuaire, de sondes, de politiques…) qui peuvent permettre de gérer plus efficacement l'exposition de services sécurisés et un contrôle de performance à l'exécution. Pour ces raisons, nous retiendrons cet ensemble d'outils pour la mise en œuvre de notre solution (voir figure 6).

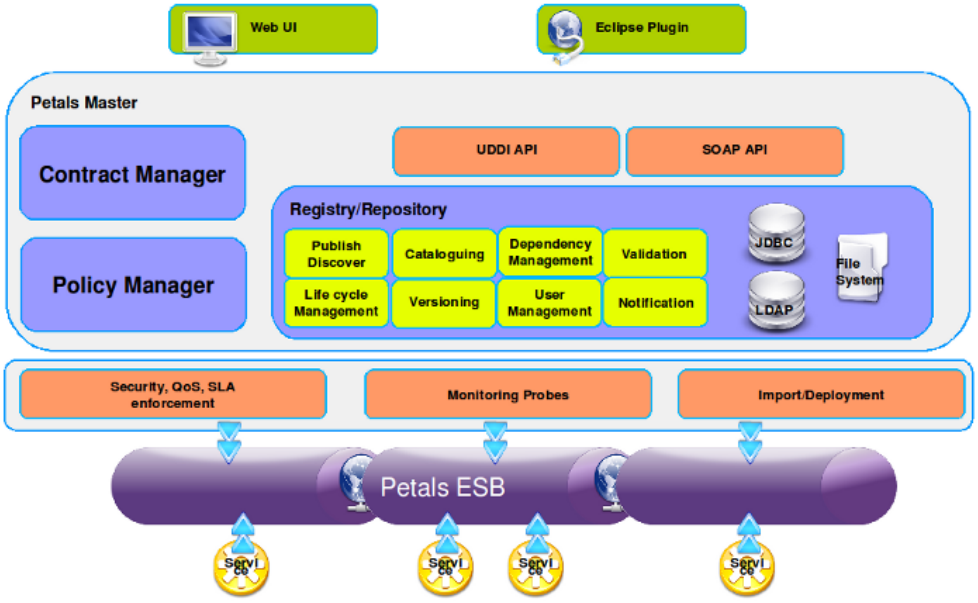


Figure 6. PEtALS Master organization [25]

Sécurité dans le SOA et le Service Web

Le modèle de référence de l'OASIS concernant les architectures à base de service identifie plusieurs facteurs clés de sécurité, comme par exemple la confidentialité,

l'intégrité, la disponibilité, l'authentification… [7]. Concernant les déploiements des architectures orientées services grâce à la technologie des services web, plusieurs standard proposent des mécanismes au niveau des messages SOAP et XML pour la sécurité, e.g. SAML, XACML, XML-DSIG, WS-Security, XML-Encryption, WS-Security, etc .

Ces protocoles de sécurité visent principalement à apporter une protection de type « point à point », c'est-à-dire permettre un échange sécurisé de ressources. Toutefois, pour garantir une sécurité de « bout-en-bout » sur l'ensemble du cycle de vie des biens, d'autres approches doivent être prises en compte.

Vers la sécurité bout-en-bout pour les systèmes collaboratif

La sécurité de « bout-en-bout » pour les systèmes collaboratifs peut être caractérisée par deux questions: « Quel partenaire peut accéder à l'un de mes biens ? » et « Que peut-il faire avec mes biens ? ».

Pour répondre à ces questions, on peut s'inspirer des systèmes de gestion des droits numériques (DRM).

DRM, contrôle d'accès : outils pour le contrôle de l'utilisation des ressources

La gestion des droits numériques peut être définie comme « la description, l'identification, la négociation, la protection, la surveillance et le suivi de toutes les formes d'usages des droits sur les actifs corporels et incorporels, y compris la gestion des relations détenteurs de droits » [11].

La figure 7 représente les fonctions les plus importantes de l'architecture DRM.

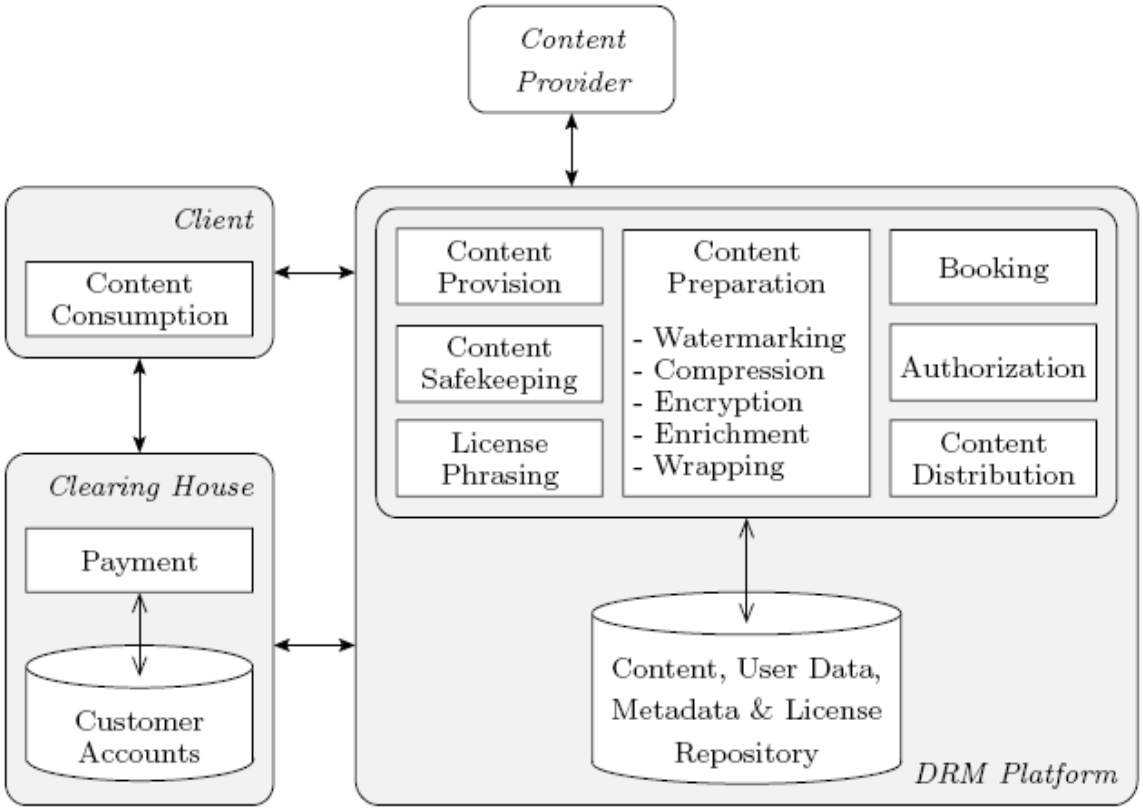


Figure 7. Un example de système de DRM

Le langage d'expression des droits (REL) est l'un des éléments fondamentaux des systèmes de DRM. Ce langage fournit un moyen pour exprimer les droits sur es contenus numériques (voir figure 8) en définissant les façons de consommer les ressources, les usages associés aux ressources, les contraintes et les obligations.

Un système de contrôle d'accès est très semblable au langage d'expression des droits pour un usage précis [29] [24] [28] [18]. Le droit d'accès est défini par un ensemble d'attributs portant sur les ressources (« Objects »), le consommateur (i.e. celui qui va demander l'accès, le « Subject ») et éventuellement portant sur le contexte (environnement, système, etc.).

Dans le contexte collaboratif, nous avons besoin de d'étendre ces modèles pour gérer convenablement des politiques de sécurité. Tout d'abord, la combinaison de

règles issues de politiques différentes puisse détecter les conflits entre règles. Ensuite, une base commune de vocabulaire est nécessaire pour décrire les connaissances sur les domaines d'application des utilisateurs (voir figure 9). Ceci améliore l'interopérabilité et une meilleur compréhension des besoins de sécurité permet d'augmenter la confiance (dans un environnement ouvert comme Internet ou semi ouvert comme une organisation collaborative interentreprises.

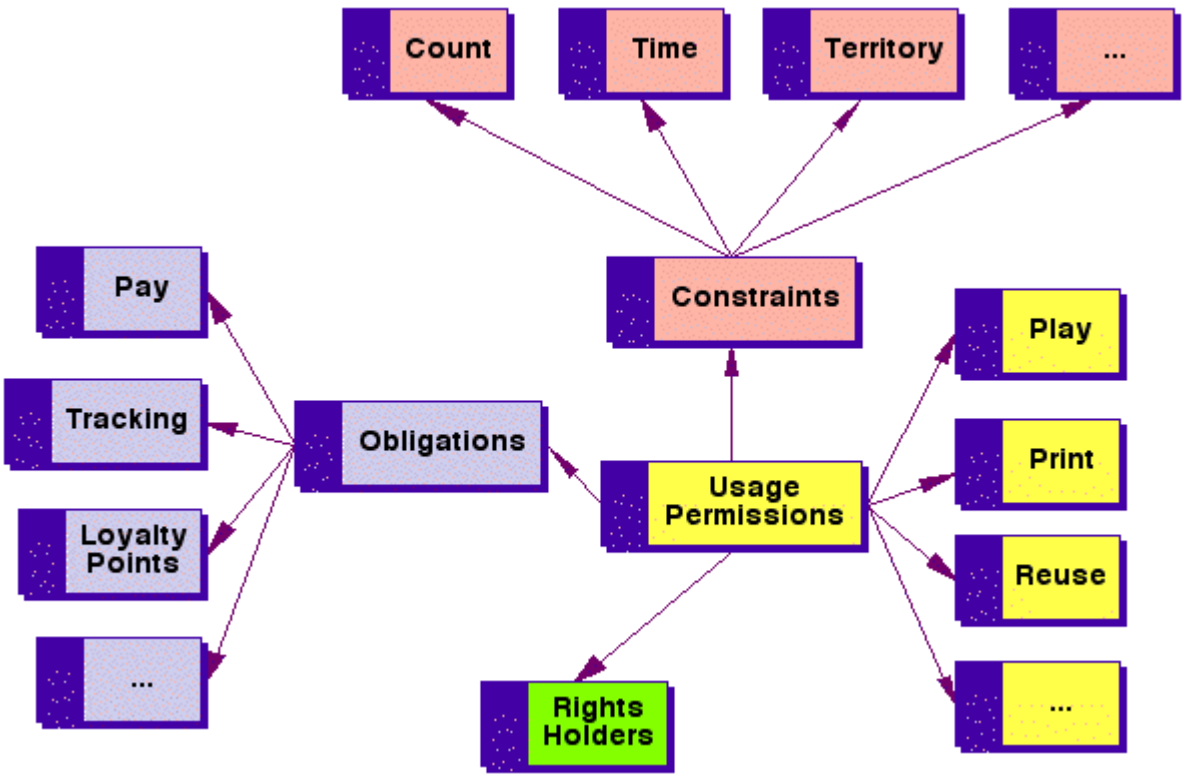


Figure 8. Un langage d'expression des droits (REL)

Basé sur l'état de l'art et à partir des besoins que nous avons identifiés (voir ci-dessus), notre travail vise à construire un système de contrôle de l'utilisation de ressources impliquées dans une collaboration pour supporter une sécurité de bout en bout.

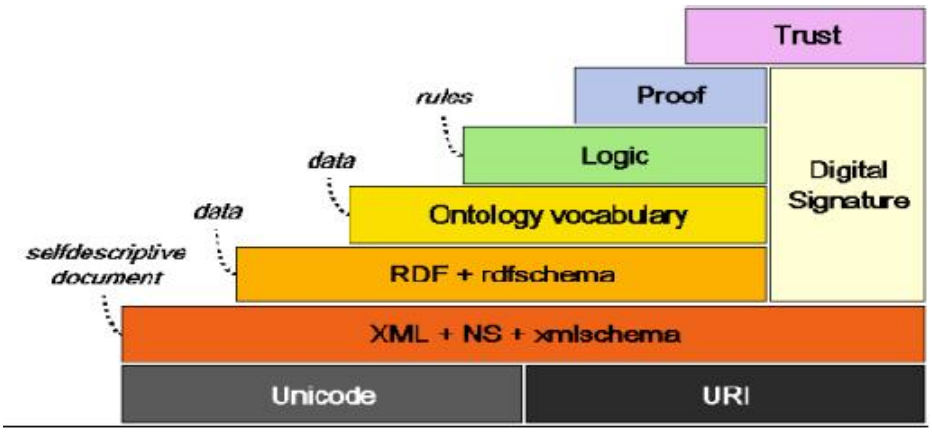


Figure 9. Modèle « Samantic trust »

Un exemple de scénario

Pour illustrer nos propositions, nous proposons d'utiliser un cas d'usage reposant sur la collaboration de trois partenaires : un établissement de soin (Bonetat Clinique), un laboratoire d'analyse spécialisé (Cardis Santé) et un assureur (Assure Direct). L'information du dossier médicale d'Alice est conservée par 'Bonetat Clinique' (partenaire B). Ce dossier inclut également des informations (Cardiac Exams) venant du laboratoire d'analyse Cardis Santé (partenaire C). Une société d'assurance 'Assure Direct' (partenaire D) consulte des informations issues du dossier médical d'Alice en faisant une requête à B. Afin de répondre D, B contacte le partenaire C pour obtenir l'information nécessaire 'cardiaques exam' puis l'intégrer avec les autres informations de son propre dossier pour pouvoir répondre à D.

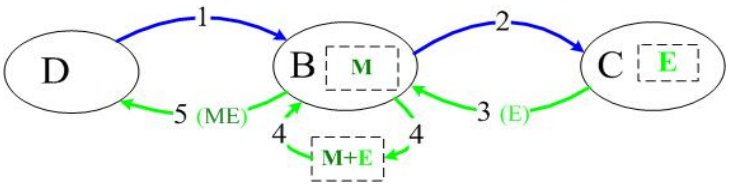


Figure 10. Un exemple de scénario

Les politiques de B et C exprimant les besoins de protection sont définies de la manière suivante :

RoP_B:

(1) Ceux qui veulent lire les informations doivent avoir une certification,

(2) Ils doivent avoir faits des accès dans les derniers 90 jours (partenaire actif),

(3) L'information doit toujours être chiffrée avec «RSA»,

(4) Le transport d'information doit être sécurisé en utilisant le protocole SSL.

Pour le partenaire C les besoins sont exprimés par la politique RoP_C:

(1) Ceux qui veulent lire l'information doivent avoir une certification,

(2) Ils doivent également avoir accédé à l'information dans les 10 jours précédents,

(3) Celui qui lit une information (de fait obtient une copie de l'information) doit la supprimer dans les 30 jours.

Dans cet exemple, les informations issues de B et C sont fusionnées à l'étape 4 (voir figure 11).

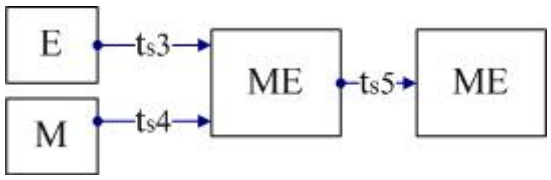


Figure 11. Fusionnement d'actif

Pour garantir une protection, il faut que les besoins de protection exprimés par un fournisseur puissent être préservés sur l'ensemble du cycle de vie des biens. Dans le cas d'un processus d'affaires avec seulement deux partenaires, le cycle de vie d'un bien ne comprend que quelques étapes bien identifiées (« production »,

« consommation » et « fin du cycle de vie »). En revanche, dans une collaboration incluant plusieurs partenaires, la situation est plus complexe puisqu'un « artefact » du processus de collaboration, c'est-à-dire un bien composite que nous appelons C-Asset, dépend du contexte de collaboration et intègre des biens « originaux » (O-Asset) provenant de nombreux fournisseurs. Comme les biens originaux (O-Asset) sont fusionnés dans le bien composite (C-Asset), ils suivent alors un nouveau « cycle de vie dérivé » par rapport à au « cycle de vie initiale » (en tant que O-Asset). Les droits des fournisseurs de ces biens doivent donc être protégés non seulement dans le cycle de vie principal du bien protégé mais aussi dans l'ensemble des cycles de vie dérivés. Par conséquent, le consommateur d'un bien composite (C-Asset) doit suivre les politiques des biens originaux (O-Assets), politiques qui doivent être agrégée dans la politique associée au bien composite (C-Asset) pour obtenir une protection de bout en bout sur la totalité du cycle de vie des biens (voir figure 12).

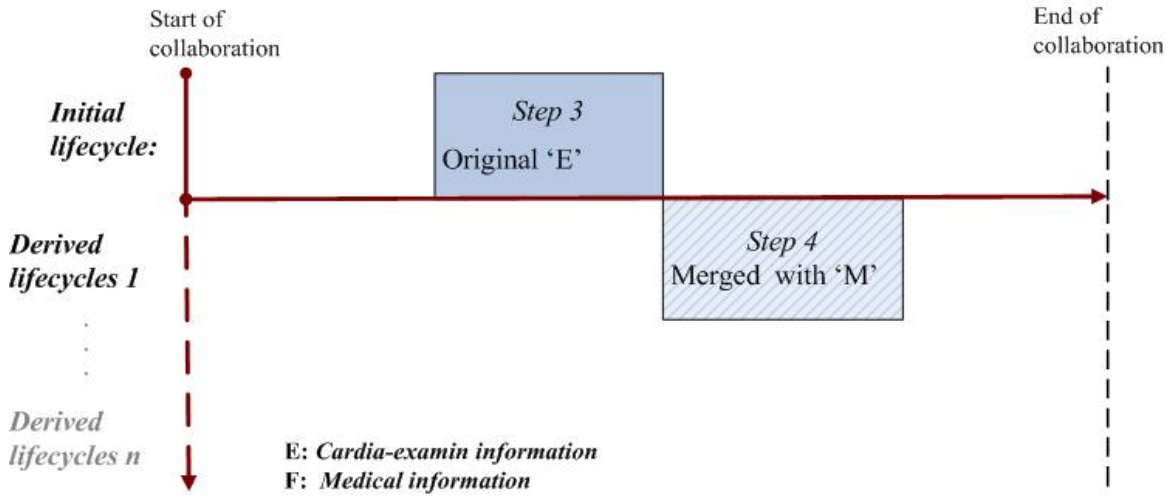


Figure 12. Cycle de vie d'actif

Dans ce qui suit, nous présentons d'abord notre architecture permettant une gestion de bout en bout de la sécurité.

Architecture générale

De manière à garantir la portabilité de notre solution, les fonctionnalités mises en œuvre dans notre architecture sont fournies par des composants exposés comme des services (voir figure 13) :

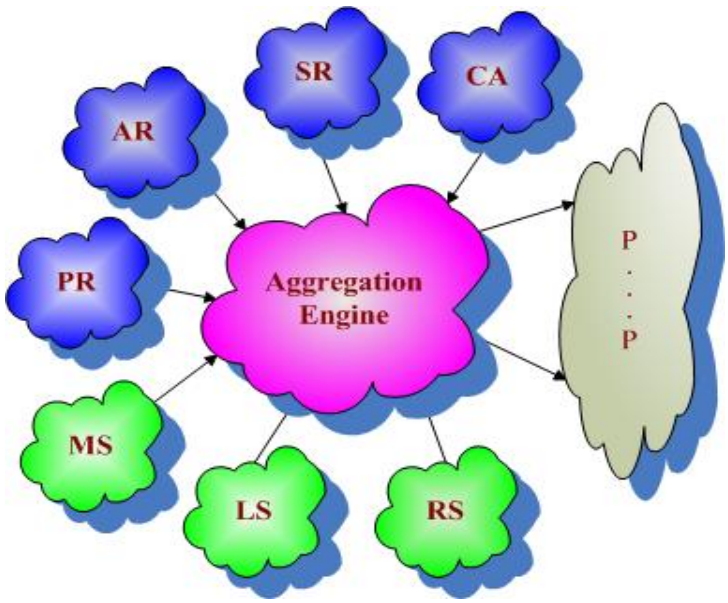


Figure 13. Architecture générale

- CA (Certificate Authority – Autorité de certification): c'est un "tiers de confiance" qui sert comme une autorité pour certifier l'identité des partnaires.

- SR (Service Registration – Annuaire): Conformément à l'architecture orienté service, ce composant est un service qui maintient l'information sur les services disponibles.

- AR (Attribute Repository – Gestionnaire des attributs) : c'est un service utilisé pour stocker et maintenir les attributs liés à la Qualité de Service (QoS) des partenaires.

- PR (Policy Registration – Registre des politiques): C'est une service permettant le stockage des politiques des partnaires.

- AE (Agregation Engine – Moteur d'agrégation): ce service offre trois fonctionnalités, à savoir un « moteur de négociation de politique », un « moteur d'agrégation de politiques » et un « gestionnaire de contexte »:

    - Le « moteur de négociation de politique » est utilisé pour décider si les deux parties peuvent travailler ensemble, l'un agissant en tant que fournisseur, l'autre en tant que consommateur.

    - Lorsque plusieurs fournisseurs d'actifs collaborent pour coproduire l'artefact final, leurs politiques sont fusionnées par le « moteur d'agrégation de politique » pour fournir une protection longue durée (pendant toute la durée de vie) sur les biens partagés et respecter les contraintes de propriété intellectuelle.

    - Le « gestionnaire de contexte » détermine les politiques qui doivent être regroupées en analysant les processus collaboratifs.

- MS (Monitoring Service – Service de monitoring): ce service de surveillance inspecte les activités liées à la consommation des biens.

- LS (Logging service): ce service enregistre les activités des partenaires pour permettre d'identifier la « provenance » des éléments du contexte et apporter la preuve du respect des politiques de sécurité.

- RS (Reputation Service – Service de réputation) : ce service est utilisée pour stocker les résultats d'évaluation venant du service de monitoring. Ces résultats sont ensuite utilisés pour évaluer la « réputation » des partenaires et améliorer le processus de sélection.

Modèle de Politiques de Sécurité

Une politique (Policy dans les équations suivantes) est construite comme une combinaison logique de règles (Rules dans les équations suivantes)

$$\text{P}olicy = ["\neg"], Rule, \{["\wedge"|"\vee"], ["\neg"], Rule \quad (1)$$

Une règle est un tuple:

$$Rule = (O, SH, S, CN, R, RN, OB, L, T) \quad (2)$$

avec (voir figure 11) :

- « SH » (Stakeholder) c'est le propriétaire de l'affirmation (règle) et c'est le propriétaire ou le copropriétaire du bien (Asset) lié à l'affirmation.
- « S » (Subject) est la partie qui peut obtenir le droit défini dans la règle sur le bien protégé.
- « O » (Object) est le bien qui est protégés par la règle.
- « R » (Right) est le « Droit » associé à la règle. C'est l'opération définie par « SH »sur le bien protégé que le sujet peut être autorisé à exercer.
- « RN » (Restriction) est une contrainte limitant l'exercice des droits.
- « CN » (Condition) est une exigence qui doit être satisfaite par le sujet pour qu'il puisse obtenir les droits sur l'objet. Cette contrainte peut porter sur trois sortes d'attributs: les attributs des sujets (SAT), les attributs des objets (OAT) ou des attributs contextuels (CNAT).

- « OB » (Obligation) est l'obligation qui doit être remplie par le sujet quand il obtient le droit.
- « L » (opérateur logique) est un opérateur logique comme "←",″∧″ et ″∨″.
- « T » (Temps) est le facteur temporel qui définit le cycle de vie de la règle.

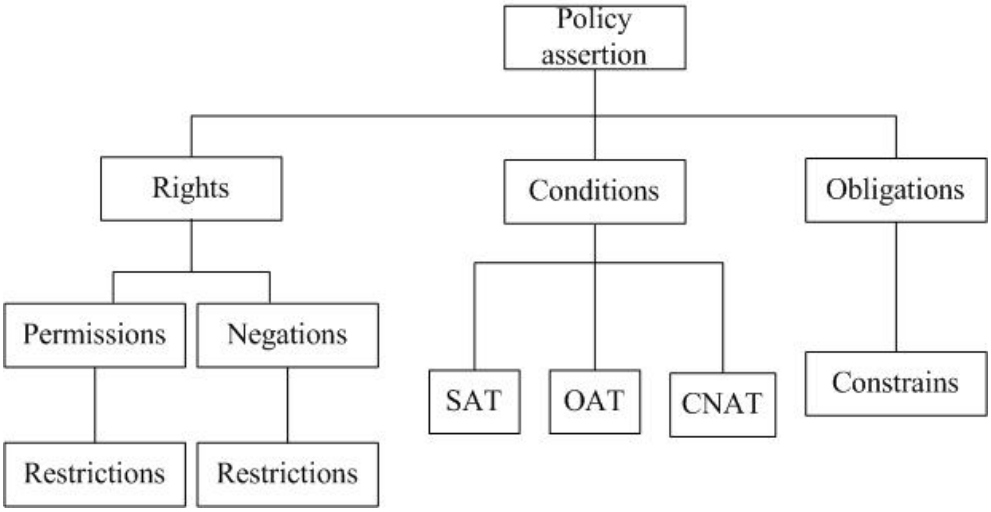


Figure 14. Eléments d'une politique de sécurité

Les règles définissant les politiques sont différenciées selon leur provenance (voir figure 12). Lorsqu'elles sont spécifiées par le fournisseur d'un bien à protéger, ces règles permettent de définir l'exigence de protection du fournisseur pour ses biens (nous les désignerons ultérieurement comme un « Requirement of Protection(RoP) »). En revanche, lorsqu'elles sont définies par un consommateur de bien, ces règles permettent d'exposer ses promesses relatives à la protection qu'il compte offrir pour ces biens et l'usage qu'il compte en faire. On parle alors de « Quality of Protection (QoP) ».

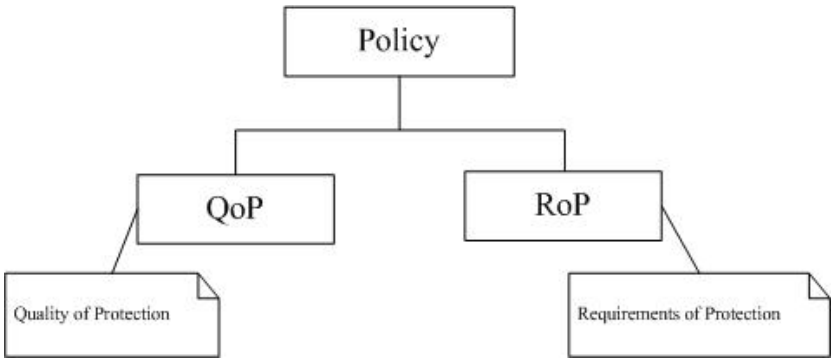


Figure 15. Modèle de politique de sécurité

Un partenaire est considéré comme un fournisseur de biens dès lors qu'il délivre des biens à d'autres partenaires intervenant dans la collaboration (peu importe qu'il agisse comme un prestataire de service ou comme un consommateur de service). A ce titre il devra définir des politiques correspondant à l'expression du RoP. Dès qu'un partenair reçoit des biens venant des autres, il devient un consommateur de biens et doit donc définir la protection et l'usage relatifs à ces biens en définissant la QoP. Notons qu'un partenaire peut agir à la fois comme fournisseur et comme consommateur de biens. Par exemple une entreprise de services de « data mining » est un fournisseur de biens (les « services d'analyses des données ») et peut donc définir des besoins de protection sur ces services. En même temps, cette entreprise est aussi un consommateur de biens puisqu'elle recevra et consommera les données provenant de son client. A ce titre, elle doit donc définir les règles permettant protéger ces données. Ainsi, la politique d'un partenaire peut avoir simultanément deux parties: RoPP et QoPP.

Dans les processus mis en œuvre dans une fédération d'entreprises, la négociation est menée entre la QoP_P du consommateur de biens (notée comme QoP_{AC} dans la suite) et le besoin de protection RoP_P du fournisseur de biens (noté RoP_{AP} dans la suite).

Selon ce modèle, les politiques des partenaires de notre exemple sont les suivantes :

$$
\begin{aligned}
&RoP'_C.(lc = eot) : \\
&\quad Rt(actionID = read \vee actionID = merge) \\
&\quad \wedge Ob(actionID = delete \wedge actionTarget = O \\
&\qquad \wedge actionTime < 30days) \\
&\quad \leftarrow \\
&\qquad Sh(C = 100)) \\
&\qquad\quad OAT(ID = E(A)) \\
&\qquad \wedge SAT(certifier = A \wedge lastAccess < 10days)
\end{aligned}
$$

Figure 16. RoP de C

$$
\begin{aligned}
&QoP_B : \\
&\quad Rt(actionID = read \vee actionID = merge) \\
&\quad \wedge Ob(actionID = delete \wedge actionTimede < 20days) \\
&\quad \leftarrow \\
&\qquad OAT(objectID = E(A)) \\
&\qquad \wedge SAT(subjectID = B \wedge certifier = A) \\
&\qquad\quad \wedge lastAccess < 9days)
\end{aligned}
$$

Figure 17. QoP de B

$$
\begin{aligned}
&RoP'_B = \\
&\quad Rt(actionID = read \vee actionID = merge).(lc = eot) \\
&\quad \leftarrow \\
&\qquad Sh(B = 100)) \\
&\qquad\quad OAT(ID = M(A) \wedge encrypted = RSA).lc = eot \\
&\qquad \wedge SAT(certifier = A \wedge lastAccess < 90days).(lc = eot) \\
&\qquad \wedge CNAT(dilveryChannel = SSL).(lc = dp)
\end{aligned}
$$

Figure 18. RoP de B

Les politiques de sécurité peuvent être définies avec différents niveaux de détail. Cela rend les langages de spécification des politiques de sécurité expressifs et

permet également de décider des règles d'arbitrage des combinaisons, c'est-à-dire définir comment les sous-politiques ou sous-règles sont constituées et comment les contradictions sont résolues. Les règles d'arbitrage des combinaisons les plus utilisées sont « deny override» (l'interdiction prime) et « permit override » (l'autorisation prime). Lors de la définition d'une stratégie d'arbitrage, il faut se montrer prudent pour éviter des « résultats inattendus » comme par exemple lorsqu'une règle « plus sure » est remplacée par une règle « moins sure ». Dans notre modèle de politique, nous avons retenu la stratégie de « deny override » comme stratégie par défaut. Ceci correspond au principe de « Negation as Failure », c'est-à-dire qu'une demande est refusée par défaut sauf si le demandeur démontre qu'il a toutes les caractéristiques nécessaires pour correspondre avec ununе règle d'autorisation (« permit rule »). A partir de ces principes et stratégie, nous construisons un modèle de politique en couche permettant de tout interdire, sauf ce qui est explicitement autorisé et pour ce dernier cas à condition qu'aucune autre règle ne vienne contredire cette autorisation.

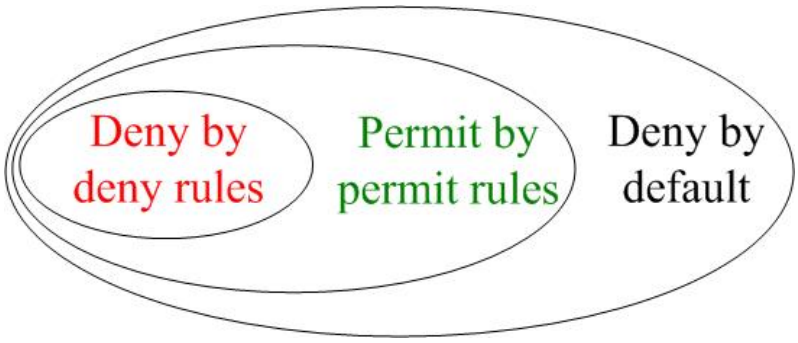


Figure 19. Organisation multi-couches de la stratégie d'autorisation

Le processus de négociation

Dans une vision globale, le modèle de négociation reflète les changements d'état de chaque participant dans le processus d'affaires. La négociation porte sur la

promesse de protection et d'usage d'un consommateur (AC) QoP_{AC} et l'exigence de protection d'un fournisseur (AP) RoP_{AP}. Le modèle de négociation permet de définir les droits, conditions et obligations portant sur des attributs du sujet (« subject »), c'est-à-dire celui qui demande le droit, de l'objet (« object ») c'est-à-dire la ressource sur laquelle on demande le droit et éventuellement des attributs liés à l'environnement. La réponse permet d'accorder ou de refuser les droits et éventuellement d'y adjoindre des restrictions et / ou des obligations. Les différentes règles sont exprimées ci-dessous.

A. La negotiation

$$[\neg]permit(R,S,O,OB,RN,T) \leftarrow QoP_{AC} \supset_{Sa} RoP_{AP} \quad (3)$$

Dans la formule ci-dessus, « R », « S », « O », « OB », « RN », « T » représentent respectivement le « droit », le « sujet », l'« objet », l'« obligation », la « restriction » et le « temps ». Cette $\supset_{Sa}$ formule (3) indique que si les promesses des consommateurs (leur QoP) respectent les exigence du fournisseur (RoP), des droits peuvent être accordés en étant éventuellement accompagnés de restrictions et obligations. « T » est le facteur temporel caractéristique du cycle de la règle dans la politique de sécurité. La fonction indique que la QoP_{AC} remplit l'exigence RoP_{AP}.

B. L'obligation

Les obligations sont considérées comme des contraintes de comportement du système dans le futur. La promesse de respecter une obligation peut être considérée comme une condition pour l'octroi des droits.

$$QoP_{AC}.OB \supset_{Sa} RoP_{AP}.OB \quad (4)$$

La formule ci-dessus indique que l'obligation figurant dans la QoP concernant la consommation d'un bien satisfait l'obligation exprimée dans la RoP du fournisseur de ce bien.

C. Requête

$$req(R, QoP_{AC}.SAT, QoP_{AC}.OAT, QoP_{AC}.CNAT, QoP_{AC}.OB, T) \quad (5)$$

Une requête déclare les attributs du consommateur (SAT), les attributs de la ressource demandée (OAT), des attributs contextuels ainsi que les droits que le consommateur demande et les obligations qu'il accepte de remplir.

D. Réponse

$$[\neg]permit(R, S, OAT, CNAT, OB, RN, T) \quad (6)$$

Une réponse peut être l'octroi de droits soumis à certaines obligations et restrictions. Dans certains cas, des attributs associés aux biens (OAT) et des attributs contextuels (CNAT) peuvent être modifiées. Une réponse peut également être la négation des droits.

Le processus d'agrégation

Le modèle formel que nous venons de définir ne décrit que la négociation entre deux partenaires. En fait, dans un contexte de collaboration, il y aura des nombreux participants. Leurs politiques doivent être comparées et combinées pour coordonner leurs exigences de sécurité. Pour répondre à ce problème, nous avons choisi d'intégrer selon une logique d'agrégation les politiques des participants dans une politique associée au contexte de collaboration (CSP) (voir figure 14).

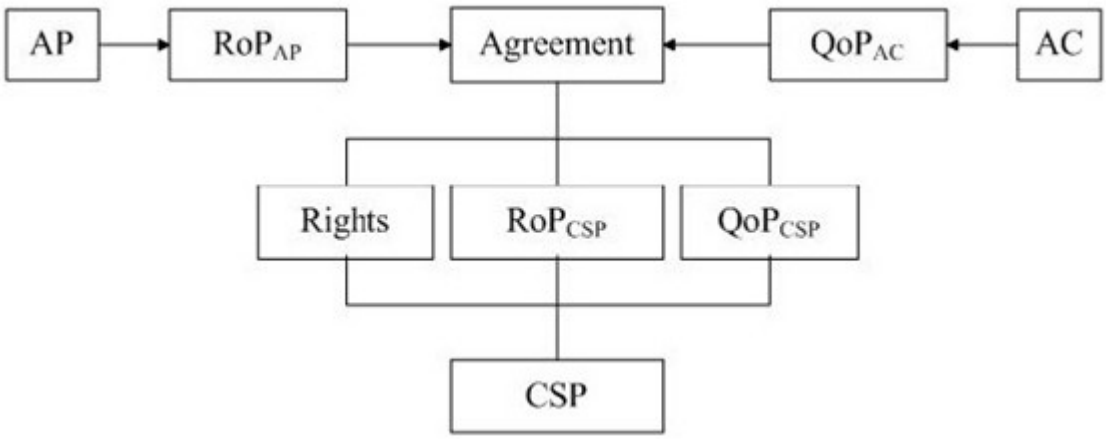


Figure 20. processus d'agrégation

A. Algorithme d'agrégation

Un contexte de collaboration, CSP comprend deux sous-ensembles de politiques représentant les exigences, RoP_{CSP} et les promesses de protection, QoP_{CSP}. La partie exigence RoP_{CSP} combine les exigences des fournisseurs (AP) RoP_{AP} et représente les droits des fournisseurs sur les artefacts liés au processus de collaboration supportant la fédération des entreprises. La partie promesse QoP_{CSP} combine les QoP_P de tous les participants. Elle représente la garantie sur les qualité de protection pour les fournisseurs qui interviendront dans le futur sur ce contexte. L'algorithme d'agrégation est défini dans les formules 7 et 8.

$$RoP_{CSP} = \sum_{i=1}^{m} RoP_{CSP} \cup_c RoP'_{APi} \tag{7}$$

$$QoP_{CSP} = \sum_{i=1}^{n} QoP_{CSP} \cup_c QoP_{Pi} \tag{8}$$

Au début de l'algorithme d'agrégation, les politiques RoP_{CSP} et QoP_{CSP} sont vides. Soit "*m"* nombre total de fournisseurs de biens dans le contexte de collaboration ,$\cup_c$

et "*n*" le nombre total de partenaires (fournisseurs et les consommateurs) dans le contexte. La fonction d'agrégation consiste à agréger les règles en résolvantles contradictions entre elles. Si une contradiction ne peut être résolue, alors les politiques des deux partenaires ne peuvent pas être agrégées et les deux fournisseurs ne peuvent pas travailler ensemble.

Avec cette méthode d'agrégation, le RoP_{CSP} calculé à l'étape 4 est le suivant :

$$
\begin{aligned}
&RoP_{CSP4}.(lc = eot) : \\
&\quad Rt(actionID = read \vee actionID == merge) \\
&\quad \wedge Ob(actionID = delete \wedge actionTarget = O \\
&\qquad \wedge actionTime = 30days) \\
&\quad \leftarrow \\
&\qquad Sh(C = 30, B = 70) \\
&\qquad\quad OAT(ID = ME(A) \wedge encrypted = RSA) \\
&\qquad \wedge SAT(certifier = A \wedge lastAccess = 10days) \\
&\qquad \wedge CNAT(dilveryChannel = SSL).(lc = dp(4))
\end{aligned}
$$

Figure 21. RoPCSP à l'étape 4

Toutefois, les participants à une fédération d'entreprises peuvent avoir des attributs de sécurité diversifiée et des politiques de sécurité également diversifiée. Ceci conduit à formuler des droits de plus en plus restreints et des conditions d'accès de plus en plus dures associés au contexte. Ainsi, il peut devenir très difficile voire impossible de trouver de nouveaux partenaires lorsque le contexte « croît ». Pour cela, nous ajoutons à notre modèle différentes stratégies d'agrégation :

• « Rich QoP_{CSP} » : cette stratégie favorise les partenaires ayant le moins d'exigences en termes de droits et offrant les contraintes les plus riches (conditions, obligations) pour permettre de répondre à un maximum d'exigence (RoP) dans le futur.

• « Slim RoP_{CSP} » privilégie les partenaires offrant un maximum de droits avec le moins de contre-parties (oligations, conditions…), c'est-à-dire avec
(1) des conditions allégées, c'est-à-dire avec moins d'éléments dans les prédicats associés aux conditions et pour ces derniers des éléments avec les plus grandes plages de valeur possibles,
(2) des " droits" les plus riches possibles exprimés dans la RoP_{CSP}
(3) des « contraintes » allégées dans la RoP_{CSP}
(5) des « obligations » également allégées dans la RoP_{CSP}
L'objectif de cette stratégie est « ralentir » la croissance de la partie exigence (RoP_{CSP}) et des promesses nécessaires (QoP_{CSP}.).

Base de connaissances

Dans notre base de connaissance, nous avons intégré des éléments relatif aux politiques (les attributes, les éléments de infrastructure de système informatique), les éléments de négociation et des éléments concernant l'application des politiques (voir figure 22).

Gestion du contexte

Pour gérer les contextes complexes, il est possible de considérer le contexte comme plusieurs sous-contextes en fonction du patron d'agrégation des biens protégés (voir figure 23).

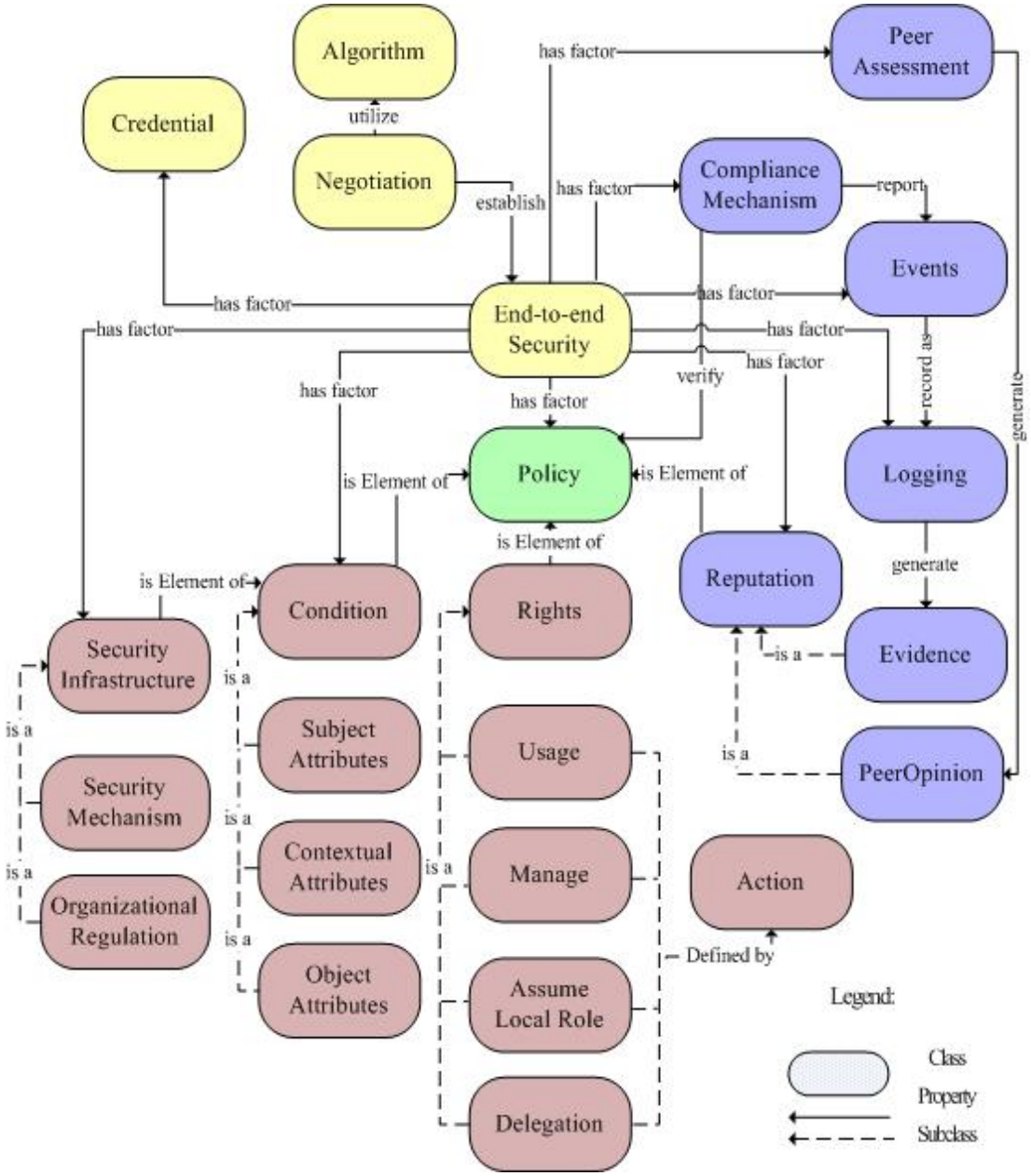


Figure 22. Ontologie globale

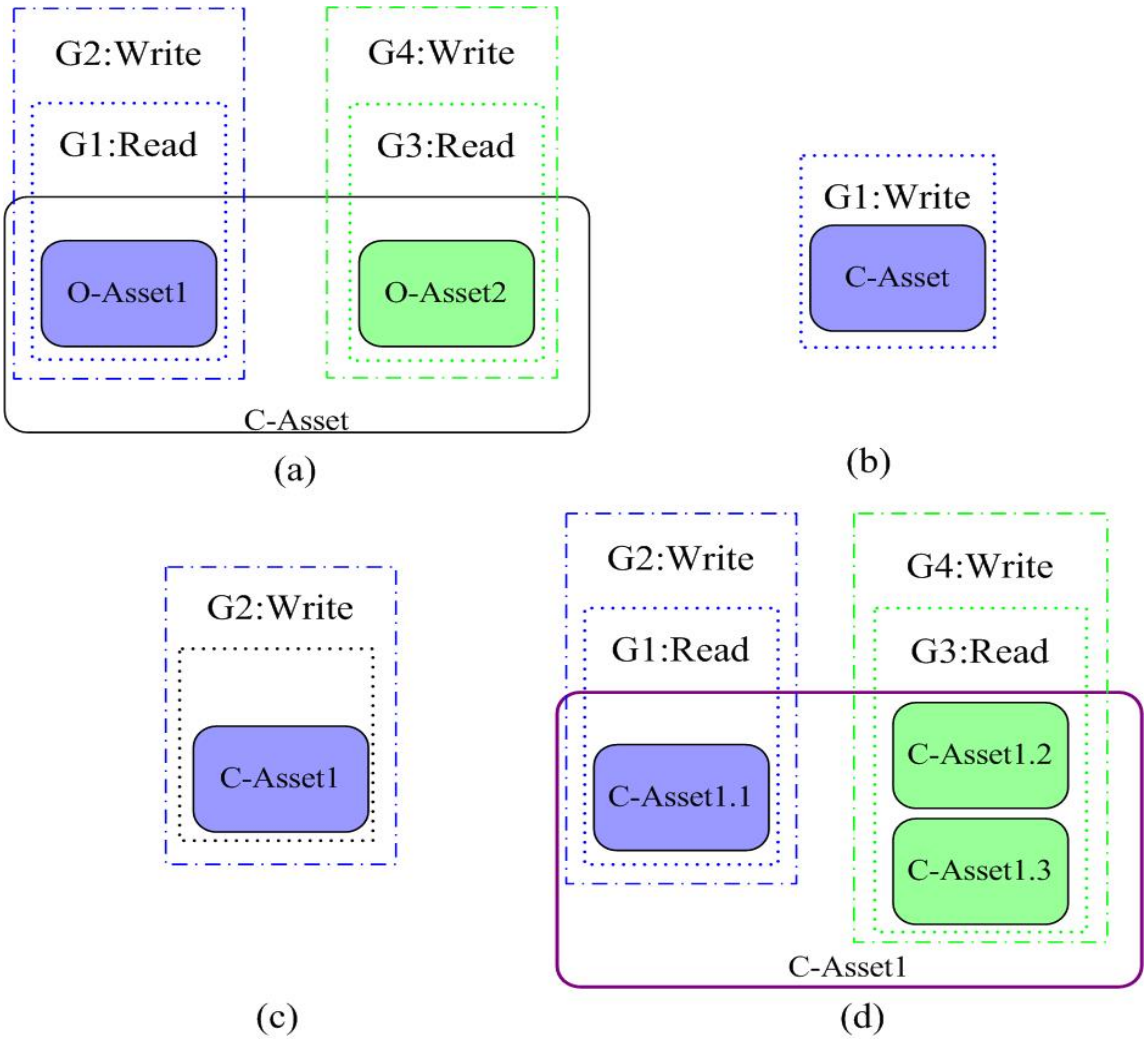


Figure 23. les modèle de sou-contexte

- EAOG : ce patron s'applique lorsque tous les fournisseurs de biens protégés peuvent distinguer leurs biens (O-assets) dans le bien composite créer (C-Asset). Dans ce cas, chaque fournisseur peut attacher ses propres politiques relatives à ses biens qui figure dans le bien composite.

- SASG : ce patron s'applique lorsque tous les biens originaux (O-Assets) sont intégrés dans un seul bien composite (C-Asset) sans qu'on puisse les distinguer et que ce bien composite suit ensuite un seul parcours (il n'y a donc qu'un seul « droit » qui s'applique). Dans ce cas, les droits sont définis en fonction des différentes politiques des fournisseurs des biens originaux.

- SAMG : ce patron s'applique lorsque tous les biens originaux (O-Assets) sont intégrés dans un seul bien composite (C-Asset) sans qu'on puisse les distinguer et que ce bien composite suit plusieurs parcours (donc plusieurs droits s'appliqueront). Dans ce cas, les politiques résultantes sont définies en fonctions des politiques originales des fournisseurs et des droits nécessaires associés à chaque parcours.

- MAMG : ce patron s'applique lorsque tous les biens originaux (O-Assets) sont intégrés dans plusieurs biens composites (C-Asset) sans qu'on puisse les distinguer et que ces biens composites suivent plusieurs parcours (donc plusieurs droits s'appliqueront). Dans ce cas, les politiques résultantes associées à chaque bien composite sont définies en fonctions des politiques des biens originaux figurant dans ces biens composites et des droits nécessaires associés à chaque parcours.

On peut trouver des exemples de la mise en œuvre de ces patrons dans un système de chaîne logistique (voir figure 24) :

- EAOG : Ce patron s'applique lorsque le fournisseur en aval (D) reçoit des informations concernant les stocks «Ia...In » des fournisseurs en amont (UP) et que ce fournisseur les combine en un seul fichier XML « I » où chaque information originale peut être identifiée. Dans ce cas, on attache la

politique originale du propriétaire UPi de l'information Ii à cette information.

- SASG : Ce patron s'applique lorsque les informations concernant l'ordonnancement de la production (« C_0a...C_0n ») venant de différents partenaires sont agrégées et utilisées par D pour les consolider dans un tableau de bord de production global (« C_4 ») en y associant un seul type d'usage.

- SAMG : Ce patron s'applique lorsque D mélange les informations concernant l'ordonnancement de la production (« C_0a...C_0n ») venant de différents partenaires pour générer un plan de production « C_5 » et associe des droits différents à ce plan (« lire », « stocker », « mettre à jour »…) en construisant des contextes différents.

- MAMG : Ce patron s'applique lorsque D réplique une information dans plusieurs « biens composites » (par exemple « C_1 », « C_2 » et « C_3 ») et que certains éléments (par exemple « C_3 ») sont dissociés des autres dans les étapes suivantes (partage avec d'autres partenaires…).

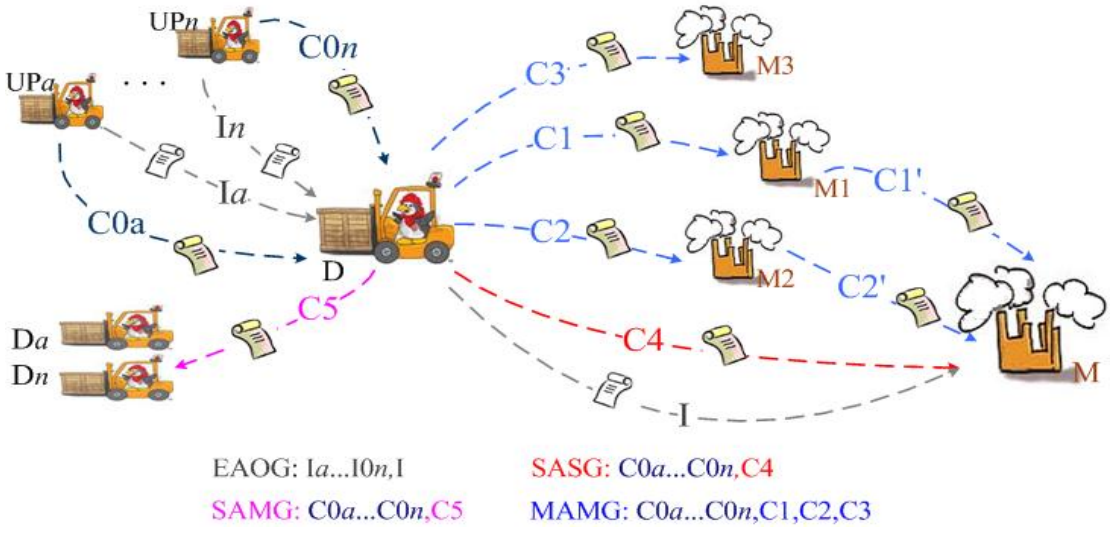


Figure 24. Exemple de modèles de sous-contextes

La mise en place de ces patrons permet de simplifier l'identification des sous-contextes. En travaillant avec des sous-contextes restreints, le nécessaire niveau de protection persiste sans que des éléments non nécessaires des politiques de sécurité soient intégrés ce qui limite la complexité globale des politiques résultantes et permet d'augmenter les chances de trouver des partenaires satisfaisant aux conditions associées au sous-contexte.

Architecture mise en oeuvre

L'architecture que nous avons implémentée dans notre prototype inclut plusieurs composants (voir figure 25) :

- Context manager : c'est le composant central pour analyser le contexte (e.g. dossier de WS-BPEL) et décider quels partenaires sont dans un même sous-contexte.
- PDP : c'est le composant permettant de réaliser la négociation des politiques des partenaires d'un même sous-contexte.
- PGP : c'est le composant qui assure l'agrégation des politiques des fournisseurs dans un même sous-contexte.
- Monitoring service : c'est le composant permettant de monitorer l'application des politiques.
- Watermarking : c'est un service permettant de tatouer les biens à protéger et donc permettre de protéger ces actifs digitaux.

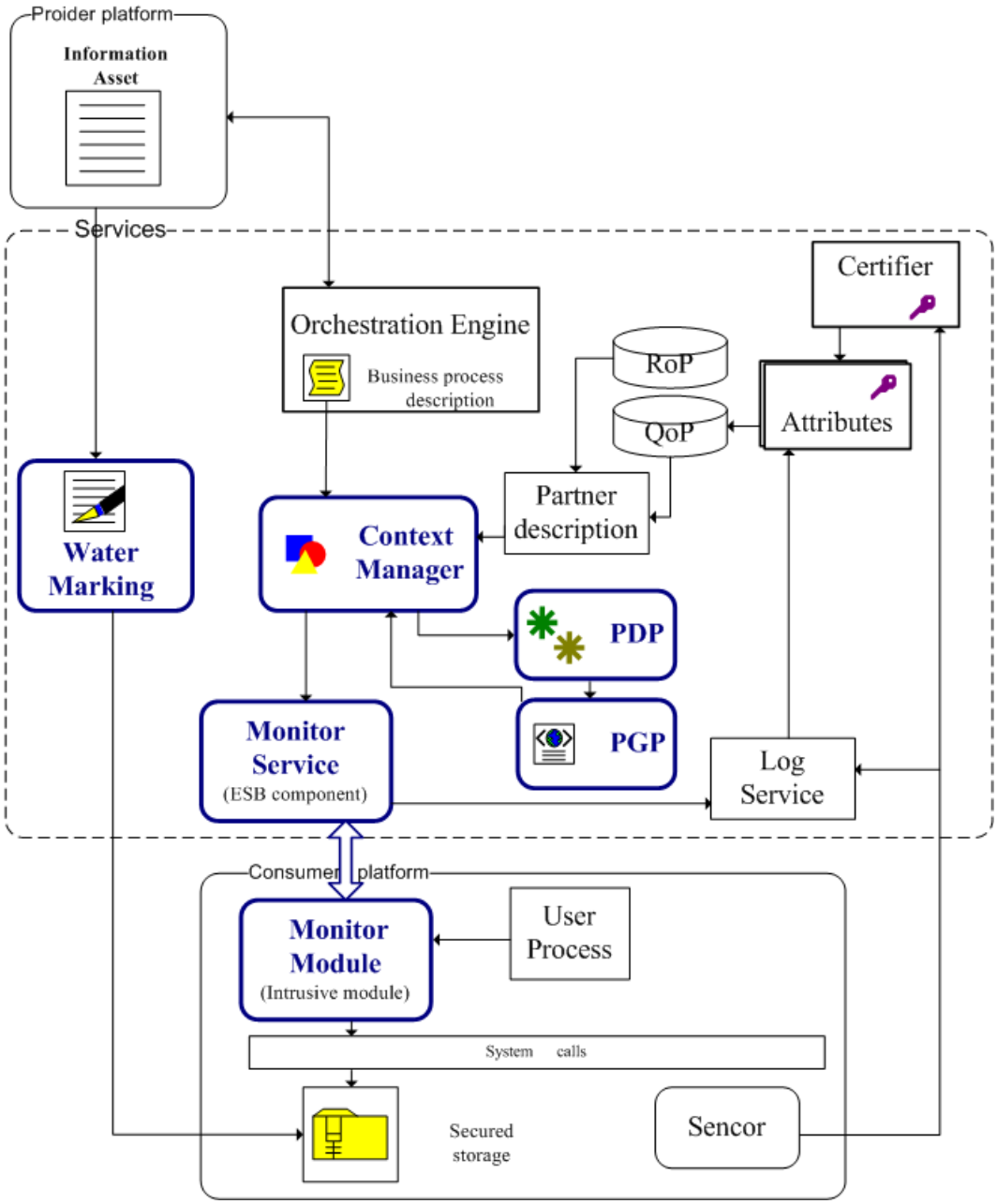


Figure 25. Architecture mise en oeuvre

Evoluation des performances

La performance du « Context manager » est testé avec les exemples de processus donnés dans la spécification « WS-BPEL 2.0 » [13]. Pour réaliser ces tests nous avons utilisé deux plateformes :

1 : Intel T7250 (Dual Core 2.0 GHz) CPU et 1.99 GHZ, 3.49G RAM (voir « time1 » dans figure 26)

2 : Intel T2330 (Dual Core 1.6 GHz) CPU et 2.00G RAM. (voir « time2 » dans cette figure)

Figure 26. Performance de « context manager »

L'évolution des performances est peu marquée selon les cas. On constate également une différence entre les environnements de test. La figure 27 montre que la consommation de mémoire reste limitée.

Memory Statistics

Filter: No filter. Click here to set filter

>Class Name	Package	Live Instances	Active Size (bytes)	Total Insta...	Total Size (bytes)	Avg. Age
CntxAnalyser	(default package)	1	224	1	224	0
Context	(default package)	12	672	12	672	0
PartnerLink	(default package)	1	32	1	32	0
PartnerLink[]	(default package)	1	16	1	16	0
Variable	(default package)	3	96	3	96	0

Figure 27. consommation de RAM

Notre composant PDP est développé avec le « SUN XACML implementation package » [26]. Ceci explique que nous ayons testé ce module avec les exemples de politique et exemples de requêtes avec les exemples fournis dans ce package (soit 4 politiques croisées avec 4 requêtes donc 16 scénarios) (voir la figure 28). On constate que les cas avec des résultats positive (P) prennent généralement moins de temps que les cas avec des résultats négatif (N).

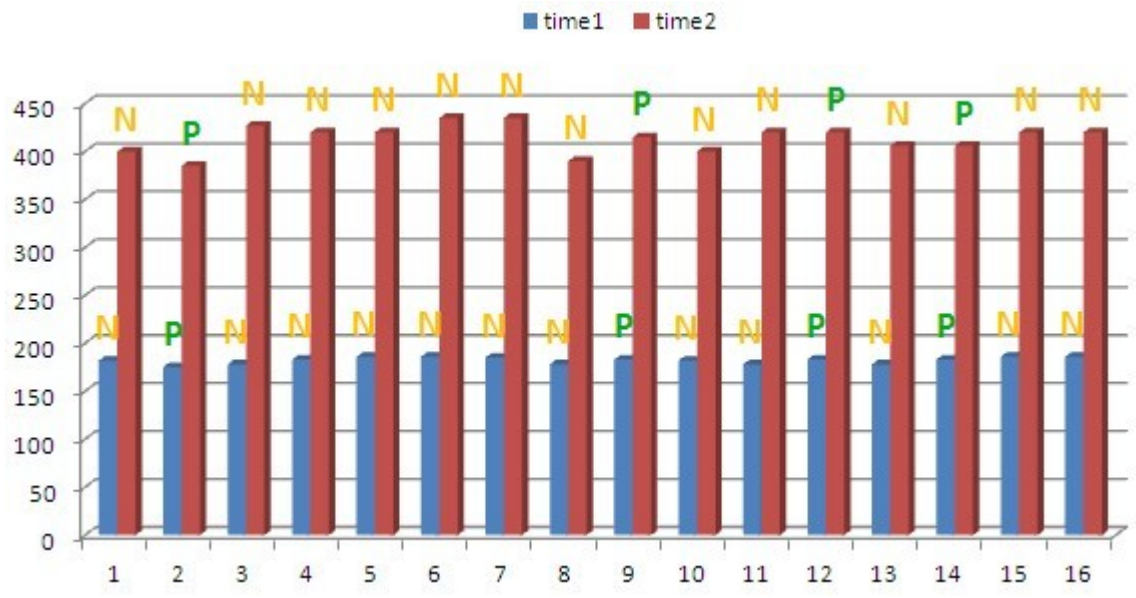


Figure 28. Performance du PDP

Notre composant « PGP » est developé avec « JAVA DOM » et « JDOM ». Nous avons testé les performances pour l'agrégation de politiques de tailles différentes

(du cas « 5*5 » correspondant à deux politiques ayant 5 prédicats chacune au cas 50*50). signifie deux politiques (rule) de 5 « predicate » (les exemple de « predicate » : « age>20 » ou « certificat=oui »). On constate que les temps nécessaires pour l'agrégation croissent avec la taille des politiques. Toutefois, les temps de traitement individuel des paires de prédicats diminuent avec l'augmentation de la taille des politiques. Ceci permet à notre module PGP de s'adapter au traitement des fichiers de politiques complexes.

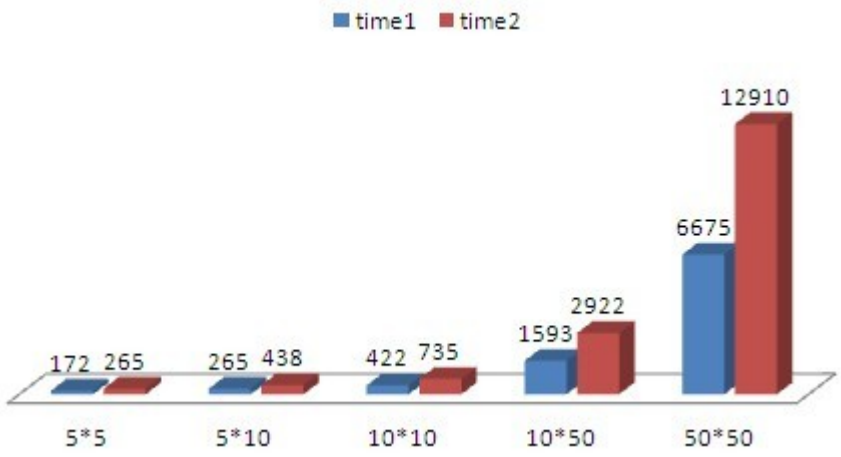


Figure 29. Performance pour l'agrégation de politiques de tailles différentes

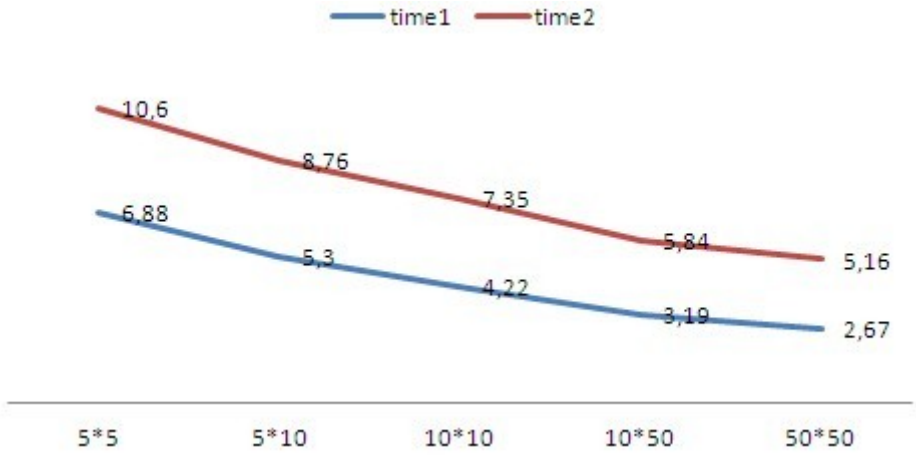


Figure 30. Performance par paire de prédicats

Conclusion

La sécurité bout-en-bout vise à protéger un actif (l'information et / ou processus, e.g. Services Web) tout au long de son cycle de vie (de la création à la destruction).

Nous avons proposé un système collaboratif de contrôle des usages pour répondre à ce cahier des charges. Pour cela, nous avons construit un modèle de contrôle d'utilisation adapté au contexte collaboratif, en fournissant une extension de syntaxe, un vocabulaire de base, un algorithme d'agrégation de politiques, une méthode de gestion du contexte et en mettant en œuvre cette architecture dans un prototype. Ce dernier nous a permis d'évaluer les performances de plusieurs composants « cœur » de notre architecture. Les travaux futurs porteront sur le développement des autres composants du système (partie « enforcement »).

Référence

[1] J. Bosak, T. McGrath, and G.K. Holman. Universal business language v2. 0, 2006.

[2] D. Box, F. Curbera, et al. Web services addressing (WS-Addressing), 2004.

[3] R. Chinnici, J.J. Moreau, A. Ryman, and S. Weerawarana. Web services description language (WSDL) version 2.0 part 1: Core language, 2004.

[4] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (WSDL) 1.1, 2001.

[5] Catteddu Daniele and Hogben Giles. Cloud Computing: Benefits, risks and recommendations for information security. Technical report, 2009.

[6] S.S.A.J.J. Dubray and MJ Martin. ebXML business process specification schema technical specification v2. 0.4, 2006.

[7] Jeff A. Estefan, K. Laskey, Francis G. McCabe, and Danny Thornton. Reference architecture for Service Oriented Architecture version 1.0, 2008.

[8] C. Evans, D. Chappell, D. Bunting, G. Tharakan, H. Shimamura, J. Durand, J. Mischkinsky, K. Nihei, K. Iwasa, M. Chapman, et al. Web services reliability (ws-reliability), ver. 1.0, 2003.

[9] H. Haas and A. Brown. *Web services glossary*, 2004.

[10] Hugo Haas. Designing the architecture for web services. *W3C*, 2003.

[11] R. Iannella. Digital rights management (DRM) architectures. *D-Lib Magazine*, 7(6), 2001.

[12] Heiser Jay and Nicolett Mark. Assessing the security risks of Cloud Computing. Technical report, 6 2008.

[13] Diana Jordan, John Evdemon, Alexandre Alves, Assaf Arkin, Sid Askary, Charlton Barreto, Ben Bloch, Francisco Curbera, Mark Ford, Yaron Goland, Alejandro Guzar, Neelakantan Kartha, Canyang Kevin Liu, Rania Khalaf, Dieter K 枚 nig, Mike Marin, Vinkesh Mehta, Satish Thatte, Danny van der Rijn, Prasad Yendluri, and Alex Yiu and. Web services Business Process Execution Language (WS-BPEL), April 2007.

[14] Lalana Kagal and Hal Abelson. Access control is an inadequate framework for privacy protection, July 2010.

[15] N. Kavantzas, D. Burdett, G. Ritzinger, T. Fletcher, Y. Lafon, and C. Barreto. Web services choreography description language version 1.0, 2004.

[16] Ban B. Linda, Cocchiara Richard, Lovejoy Kristin, Telford Ric, and Ernest Mark. The evolving role of IT managers and CIOs–findings from the 2010 IBM global IT risk study. Technical report, 2010.

[17] C.M. MacKenzie, K. Laskey, F. McCabe, P.F. Brown, and R. Metz. Reference model for service oriented architecture 1.0, 2006.

[18] OASIS. eXtensible Access Control Markup Language (XACML) version 2.0.

[19] OASIS. Universal Description, Discovery, and Integration (UDDI)3.0.2, 2004.

[20] M. Papazoglou and D. Georgakopoulos. Introduction to a special issue on service oriented computing. *Communication of the ACM*, (10):25–28, 2003.

[21] Mike P. Papazoglou and Willem-Jan Ven Den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, March 2007.

[22] M.P. Papazoglou. Extending the service oriented architecture. *Journal of Business Integration*, (10):18–21, 2 2005.

[23] M.P. Papazoglou and J. Dubray. A survey of web service technologies. Technical report, University of Trento, 2004.

[24] Jaehong Park and Ravi Sandhu. The UCON_ABC usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.

[25] PEtALS. Petals master, soa governance solution.

[26] SUN microsytem. The SUN XACML implementation.

[27] A. Tsalgatidou and T. Pilioura. An overview of standards and related technology in web services. *Distributed and Parallel Databases*, 12(2):135–162, 2002.

[28] Xinwen Zhang, Masayuki Nakae, Michael J. Covington, and Ravi Sandhu. A usage-based authorization framework for collaborative computing systems. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 180–189, New York, NY, USA, 2006. ACM.

[29] Xinwen Zhang, Masayuki Nakae, Michael J. Covington, and Ravi Sandhu. Toward a usage-based security framework for collaborative computing systems. *ACM Trans. Inf. Syst. Secur.*, 11:3:1–3:36, February 2008.

Publication

[1] Ziyi SU and Frédérique BIENNIER. A Collaborative-context Oriented Policy Model for Usage-control in Business Federation. URKE 2011. 4-7 Aug. 2011. Bali, Indonesia.

[2] Ziyi SU and Frédérique BIENNIER. Toward Comprehensive Security Policy Governance in Collaborative Enterprise. APMS2011. 26-28 Sept. 2011 Stavanger, Norway.

[3] Ziyi SU and Frédérique BIENNIER. Full Lifecycle Resource Protection in Composite Web Service with XACML. ICIC Express Letters, Part B: Applications. ISSN 2185-2766. Vol. 2, Issue 5, p. 1045-1050, October 2011.

[4] Ziyi SU and Frédérique BIENNIER. Full Lifecycle Resource Protection in Composite Web Service with XACML. ICIT2011. 12-14 Aug. 2011. Changchun, China

[5] Ziyi SU and Frédérique BIENNIER. End-to-end Security Policy Description and Management for Collaborative System. Journal of Information Assurance and Security, 2011.

[6] Ziyi SU and Frédérique BIENNIER. End-to-end Security Policy Description and Management for Collaborative System. Proceeding of the IAS2010, Aug. 23-25, 2010, Atlanta, USA.