uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Waël Hassan

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)

GRADE / DEGREE

Department of Computer Science

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Validating Legal Compliance – Governance Analysis Method

TITRE DE LA THÈSE / TITLE OF THESIS

L. Logrippo

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

L. Korba

L. Peyton

S. Matwin

M. Weiss

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Validating Legal Compliance

Governance Analysis Method

by

Waël A. Hassan

Supervisor: Luigi Logrippo

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfilment
of the requirements for the degree of

Ph.D. in Computer Science

Under the auspices of the
Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering (SITE)
University of Ottawa
Ottawa, Ontario, Canada

Your file Votre référence
ISBN: 978-0-494-59486-5
Our file Notre référence
ISBN: 978-0-494-59486-5

Abstract

This dissertation introduces a logic-based, computer assisted framework for validating legal compliance of enterprise governance models. This framework is intended to help checking whether governance systems are consistent with the law. The framework proposes legal and enterprise models, a governance analysis method - called GAM -, in addition to a governance analysis language - called GAL-, and an implemented governance analysis tool - called GAT. The method consists in extracting requirements into GAL statements by using patterns and translating them into a logic model for consistency checking.

The models, language, and tool proposed in this thesis evolved as a result of their application to governance laws (privacy and financial). The method's main processes were validated through application to Canadian and US laws (mainly PIPEDA and Sarbanes-Oxley) combined with various examples taken from enterprise systems. From these evaluations it was concluded that the method is formal and repeatable for the purposes of partially extracting requirements from laws and enterprises, directly mapping these combined requirements into logic models, and checking results.

The extraction process uses patterns to match legal and enterprise requirements. The representation process maps extracted requirements to GAL statements. The generation process takes as input GAL statements to generate a logic model . The Alloy logic analyser is then used to check legal consistency. Three legal-compliance validation techniques are proposed: model, ontology, and scenario checks. Model-checks validate the combined legal and enterprise requirements for logical consistency. Ontology-checks validate the enterprise structure and process. Scenario-checks validate enterprise scenarios.

The techniques are shown to be useful for identifying conflicts and extracting counterexamples.

Acknowledgements

First and foremost thanks to my mentor and supervisor Prof. Luigi Logrippo. Without his recommendations and suggestions this work would have not been completed. I also like to thank him for his patience and diligence. I like to thank Jacques Sincennes and Prof. Daniel Amyot for their discussions and recommendations throughout the years.

I like to thank all of those in the open-source community for their contributions to the languages and tools used in this thesis. These tools include: Alloy, ArgoUML, GraphViz, KodKod, MjoGraph, MindNote, and Open Office.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AR	Access Requestor
CEO	Chief Executive Officer
CFO	Chief Financial Officer
CGO	Chief GO
CIO	Chief Information Officer
CP	Companion Policy
CPO	Chief Privacy Officer
CSA	Canadian Standards Association
CTO	Chief Technology Officer
ECA	Event Condition Action
EU	European Union
FOIPPA	Freedom of Information and Protection of Privacy Act
GAL	Governance Analysis Language
GAM	Governance Analysis Method
GAT	Governance Analysis Tool
GO	Governance Officer
HIC	Health Information Custodian
HINP	Health Information Network Provider
IETF	Internet Engineering Task Force
IFRS	International Financial Reporting Standards
ISO	International Standards Organisation
MFIPPA	Municipal Freedom of Information and Protection of Privacy Act
NGO	Non Governmental Organisation
PAP	Policy Access Point
PCAOB	Public Company Accounting Oversight Board
PDP	Policy Deployment Point
PEP	Policy Enforcement Point
PHIPPA	Personal Health Information Protection Act
PIP	Policy Information Point
PIPEDA	Personal Information Protection and Electronic Documents Act (The Act)
PO	Privacy Officer See CPO
RAP	Requirements Access Point
RCP	Requirements Compliance Point
RFC	Reference For Comments
RIP	Requirements Information Point
RTP	Rule Translation Point
RVP	Requirements Validation Point
SEC	Securities and Exchange Commission
SOX	Sarbanes-Oxley Act
UK	United Kingdom
UML	Universal Modeling Language

To Ali and Fatmé

Chapter 1

Legal Compliance

1.0 Introduction

Legal systems are traditionally expressed in natural language [Lodder 2008]. Increasingly however, laws include norms created with the intention of determining directly or indirectly the functioning of computing systems. These norms must be translated into a formal language and implemented in software, which is, directly or indirectly, executable by computer programs. Ideally, an objective and repeatable translation mechanism and a validation method capable of reflecting these changes in the law should exist to facilitate the process.

Translations between natural and formalised languages present well-known challenges [Cohn 2007][Gamut 1991][Pinkal 1993]. For instance, most natural language text can be translated, but many assumptions must be made. It is also conceivable that some laws, including computer code, be adopted. However, this is still a very rare practice.

Enterprises have an obligation to adhere to norms while minimizing compliance costs [Anand 2007]. This is one of the reasons for seeking automated, legal compliance tools to facilitate their compliance process [Cornelius 2006]

[Zelkowitz 2001] [Jajodia 2008] [Ramos 2006] by reducing the manual effort and eliminating risks.

One of the primary concerns of enterprise executives is compliance with governance laws including privacy and financial laws. Executives in e-business companies have been specifically affected by increasingly stricter laws and standards [Dalton 2008] [Qin 2005] [Wright 2000]. Compliance to laws is well enforced in established areas, such as the manufacturing, pharmaceutical and health-care industries. In these areas, internal auditors carry legislative compliance duties in addition to financial compliance.

Compliance audit methods require the availability of effective tools for studying evolving laws [Kontogiannis 2007] [Subirana 2006]. Automated legislative compliance tools answer how the policies might affect the business-process, ranging from who has access to trade transactions, to which business partners cannot execute a taxation audit, and how long are we able to retain client information. Compliance tools are complex to design and difficult to automate, because they must be able to implement financial, security and privacy laws in addition to upholding the integrity of the critical business applications [Jajodia 2008] [Ramos 2006].

1.2 Preliminary definitions

We will follow the spiral approach [Newkirk 1988] in defining key concepts. Based on this technique, we will present basic informal definitions in Chapter 1 while continuously adding further detail in the subsequent chapters. We believe

that this approach will re-enforce concepts often confused in new research domains. We will provide informal definitions for some of the key concepts such as corporate governance (process and organisational structure), enterprise and legal governance requirements and compliance validation goals such as legal consistency and completeness.

Corporate Governance: Informally defined, corporate governance is the system by which business corporations are directed and controlled. Corporate governance has several sub-processes, makes use of tools and methods, and dictates a certain organisational structure.

This structure specifies the distribution of rights and responsibilities among different participants in the corporation and spells out the rules and procedures for making decisions on corporate affairs. It also provides the processes through which a company's objectives are set, the means for attaining those objectives and monitoring the performance.

Corporate governance system includes process and organisational structure definitions. Governance processes may include policy management, issue and risk management, as well as compliance, while organisational structure artefacts include roles performing governance aspects such as the CFO, CEO, and the GO.

Please note that we may refer interchangeably to enterprise, business and corporation.

Governance Officer (GO): The governance officer or GO is an important actor in the enterprise governance structure. This enterprise role is assigned to an individual accountable for validating legal compliance.

The GO's main responsibility is using compliance tools to validate financial, privacy and other regulatory requirements.

Governance Requirements: Informally defined, a requirement is a documented need of what a particular product or service should be or do [Fajardo 2007]. In other words, it is a statement that identifies necessary attributes, capabilities, characteristics, or qualities of a system for it to have value and utility to a user [Evers 2008]. In the classical engineering approach, sets of requirements are used as inputs into the design stages of product development. Requirements show what elements and functions are necessary for a particular project. In this thesis, we will often refer to either legal or enterprise governance requirements.

Legal and enterprise requirements are composed of several elementary requirements, because laws may have many provisions and each can be represented by one or more requirements. Similarly, an enterprise may have many business policies, which in turn can be represented by one or more requirements.

Legal and enterprise requirements include logical assertions; we will study these types of requirements extensively in Chapter 2.

Ontology: Legal and enterprise requirements are rich in ontology specifications. Ontologies are a fundamental concept in our method and will be further explained in the literature review of Chapter 3. An ontology is defined *as a formal representation of a set of concepts within a domain, together with the relationships between these concepts* [Sinha 2006]. Ontologies provide explicit models of shared conceptualizations and are useful for modeling formal, semi-formal and informal knowledge needed for communication within a company [Nikodemus 2005].

Enterprise Legal Consistency: Enterprise legal consistency, which we call legal consistency throughout this thesis, is an important concept. Legal consistency is defined as not having multiple verdicts for the same case [Simmons 2007]. Inconsistency [Wintgens 1998] is also defined as having two rules that contradict each other. Other common definitions of consistency refer to "treating similar cases alike" [Lanni 2006]. In the enterprise context, legal consistency refers to "obedience to the law" [Colby 1987]. We adopt the following definition:

Enterprise requirements are legally consistent if they adhere to the legal requirements and include no contradictions.

Enterprise Legal Completeness: Enterprise policies are said to be legally complete if they contain no gaps in the legal sense [Armour 1999]. Completeness can be though of in two ways [Armour 1999]: Some scholars make use of a concept of 'obligational' completeness such as Ayres and Gertner [Ayers 1992]. According to this usage, a system or a contract is 'obligationally' complete if it

specifies what each party is to do in every situation, even if this is not the optimal action to take under some circumstances. Others discuss 'enforceability' completeness in the sense that failing to specify key terms can lead a court to characterise a system as being too uncertain to enforce (May & Butcher v the King 1934) [Waddams 2005], and hence a system may be complete with respect to enforceability. This leads to the following definition:

enterprise regulations or requirements are legally complete if it specifies what each party is to do in each situation while covering all gaps in the legal sense.

We will be studying the problems of compliance, including consistency. We will present a limited study of completeness, with methods derived from mathematical logic.

Consistency: Informally defined, a set of requirements belonging to a single system, represented in a set of formulae, is said to be consistent if it includes no contradictions. Moreover, two systems are said to be consistent if the combination of their requirements produces no contradictions. One of these systems can include requirements from the law; the other can include enterprise requirements.

Completeness: We will define completeness since it might prove helpful for the understanding and discussion of subsequent chapters. Informally defined, *enterprise requirements are said to be complete with respect to the law if there*

are no valid scenarios under the law that cannot be represented by enterprise requirements.

Compliance: Validating enterprise compliance means validating enterprise requirements against the legal requirements. Informally defined, *enterprise requirements are said to be compliant if they satisfy two properties with respect to the law: the first is legal consistency, the second is legal completeness.* This definition will be further revisited in Chapters 2 and 3.

Compliance validation belongs to the governance processes as mandated in the regulated domains and aims to ensure that a corporate governance system is compliant with the applicable laws and regulations. Examples of regulated domains include the energy production and financial industries, the stewardship of the environment, production of food, protection of workers and the use of medical devices. Of prime importance in these regulated areas is the protection of clients' rights, health and safety.

Once laws are passed and regulations are enacted, governance officers verify the implementation of these laws and regulations. In the industrial sense, compliance audits ask whether a product has specific physical characteristics and is manufactured through specific processes. The issuance of a certificate of compliance says that rules were followed in making the item [Arter 2002]. In other words, a compliance audit seeks to validate a set of requirements or rules even where they may not necessarily be questioned. Examples of compliance audits could include tax and financial audits, for instance. Audits are helpful in

providing businesses with the assurance that their balance sheets and income statements data and processes are accurate.

Compliance validation can either be manual or automated. Automated compliance validation includes methods and tools capable of implementing legal and enterprise artefacts in addition to providing a computer representation able to partially answer the compliance questions. Further components may include issue and risk management. Automated governance methods can detect violations at the enterprise definition stage and may also include tools to analyse the activities and detect violations once they occur. Usually it is impossible to check all aspects of legal compliance in a purely automated way.

Corporate requirements are said to be compliant if they are consistent and complete with respect to the law. The validation of compliance is an enterprise governance process that occurs after the legal requirements have been captured. This can be achieved during the construction of the enterprise. However, once the enterprise is put into operation, any changes to the enterprise or legal requirements need to be revalidated.

During enterprise inception, a governance officer checks whether the enterprise system being defined adheres to the legal requirements. The governance officer (GO) can perform an a priori validation of the compliance. Alternatively, the governance officer can periodically run traceability audits through a posteriori audit. However, such an approach can be costly, for two reasons. First, it is harder to implement because it involves transaction audits. Second, it is

deployed after the problem occurs and the damage to reputation or theft has already occurred. Because laws and regulations represent contractual agreements between the enterprise and the jurisdiction, failure to meet legislative requirements will have potentially serious implications because laws have restrictions, including penalties for violations.

Compliance Data: We define compliance data as all data belonging or pertaining to an enterprise or the law that can be used for the purpose of implementing or validating compliance. It is the set of all data that is relevant to a governance officer in a court of law for the purposes of validating consistency, completeness or compliance.

1.3 Research challenges

In the traditional sense of governance, the process of legal compliance suggests a multiple step process.

1. The first step is the extraction of legal requirements. This involves an interpretation process of the legal text.
2. the second step is the assembly of compliance data. This process lists enterprise data, including processes, structures, and generic requirements for compliance.
3. the third step is validating compliance based on legal and enterprise compliance data.
4. the fourth step is the continuous monitoring for violations. Changes in the enterprise governance definition or the law would re-initiate the cycle.

Below, we will list some examples related to each of the research challenges.

1.3.1 Requirement definition challenges

Defining requirements is a challenging task because laws are written in natural language and are open to interpretation owing to interdependencies and a possible lack of precision [Belarus 1997]. Laws may declare facts, such statements are generic, hence finding a precise representation of such statements is not may not be possible. E.g. Privacy should be respected, is a generic statement. An example of a declarative requirement is the Accountability-Principle in the Canadian Personal Information Protection and Electronic Documents Act (PIPEDA) [PIPEDA 1999], an enterprise is accountable for privacy violations, and should attempt to preserve the privacy of its clients. It is possible to interpret this into: an organisation shall designate an individual or individuals who are accountable for the organisation's compliance process; the compliance process validates the privacy of the individuals after each transaction. This can be further implemented in more granular statements, if an employee requests access to clients' data without a valid purpose, then block access; if someone requests the name of the accountable individual, then deliver the name of the CPO; the CPO manages the privacy audit process in addition to the privacy reporting process

Laws can assist in defining enterprise requirements. They usually define the 'what' aspect of a requirement, whereas enterprises usually define the 'how'.

Nevertheless, the task of representing requirements is not straightforward, because it necessarily requires subject matter expertise.

1.3.2 Difficulty of checking consistency

Checking the consistency of requirements is a challenge. Given an enterprise requirement, a governance officer must ensure that its combination with the legal requirements does not create conflicting situations. The number of situations or scenarios can be considerably large, given all the possibilities, making the problem computationally difficult. For example, a requirement taken from the United States financial reporting law -Sarbanes-Oxley (SOX) [SOX 2002] section 404- states that approvals cannot be granted to transactions initiated in other departments. This requirement can be represented through a rule, such as, if the initiator is in a different department, then deny access to the approval action. As a result, an employee who works in two departments may be prevented from approving any transactions in either department. Another conflicting scenario would occur in the scenario of employee exchange between departments; there are possibly others leading to interactions.

Another example from the financial world is the SEC amended Rule 12d2-2: The delisting of securities is effective 10 days after the Form 25 is filed. This clause specifies a requirement to which an enterprise process needs to adhere. Such a property can be implemented through a function that is triggered, based on an activity such as filing Form 25.

1.3.3 Challenge in defining ontology

The definition of various enterprise ontologies is an important element for enterprise governance. An enterprise hosts an organisational structure, a processes and a possible legal ontology. Laws define ontology, including the definition of the ontology's basic classes, in addition to relating instance classes to others. Laws may specify goals in addition to process structures. In the previous subsection, the law specified a property labeled "collecting for a purpose." It dictates that data must be collected for a purpose. The law may specify a method to achieve such a property. For example, a periodic investigation activity included in the privacy process justifies the need to retain information. This interpretation requires the inclusion of an activity in the process ontology.

The law presents many ontology definitions related to a process, organisational structure and legal terms. The complexity of the requirement interpretation creates a further challenge in the refinement process. For example, in the financial reporting requirement of the previous subsection, the law dictates the process of producing a financial report by specifying resources, processes and steps involved.

1.3.4 Difficulty validating completeness

Once requirements are translated into implementations, the challenge is to investigate whether the translated rules are complete with respect to the intent of the law.

Each of the legal requirements may exhibit certain properties that need to be preserved and thus included in the enterprise model. For instance, from the personal Information Protection and Electronic Documents Act, personal data must always be collected, distributed or retained for a purpose. Such a property needs to be maintained.

In another example, the PIPEDA requires that all collected data should be used solely for the purpose for which they were collected. This requirement can be translated into: collect data for a purpose, restrict access to data unless purpose is valid , destroy data once purpose is achieved.

The check of whether the above translation really covers all possible scenarios allowed by the law is a challenge.

1.4 Research goals

Under the domain of corporate governance compliance validation, we propose a framework for validating the corporate compliance to the legal requirements. Our framework is based on a defined model, method and tool. The framework provides several features, notably a compliance method with an associated tool that allows an enterprise governance analyst to manually extract and formally

validate requirements. These features will assist an enterprise governance officer (GO) to receive assured partial confirmation of enterprise compliance.

In detail, our semi-automated process would alleviate the manual work required on each occasion an enterprise or legal requirement is amended. In addition, the validation will help in the detection of potential conflicts between requirements and show whether the legal properties are respected.

Finally, our process will help determine whether the enterprise follows the legally specified process. Our study will be restricted to the consistency aspect of compliance; the issue of completeness will be left to future work.

1.5 Research hypothesis

Corporate governance compliance requires a method to assist in validating legal compliance. Given the current research and background in enterprise governance, our research hypothesis states:

Conjecture 1 :*It is possible to partially automate the process of compliance validation of enterprise requirements to legal requirements through the use of logic-based models.*

Conjecture 2: *By using logic analysers, one can partially validate the compliance assurance of enterprise requirements to laws.*

1.6 Contributions

In this thesis we develop a compliance validation framework, which includes:

1. a meta-model
2. a high-level method
3. and an accompanying language and tool to validate consistency.

This framework is aimed at defining a governance framework for enterprises wishing to validate their governance compliance. Once implemented, the framework should define how enterprises model their compliance data. The models in the framework need to be extensible.

The framework needs to provide a practical and repeatable method to validate the compliance to law. In addition, the framework will include a language and tool (a prototype) that can be used to for consistency checking. It will also be able to resolve governance laws of different types. Canadian and United States laws will be presented in the case study.

This framework is aimed at defining a process for enterprises wishing to validate their governance compliance. Once implemented, the framework should define how enterprises model their compliance data. The models in the framework need to be extensible.

1.6.1 Meta model

The proposed meta model serves as a common base model to describe the enterprise and legal requirements and define the syntax and semantics of the governance requirements. The meta model must also define the semantics of the proposed artefacts in the framework. Once instantiated, the enterprise model

can fit any organisation. Moreover, the model needs to be able to represent the patterns used in the legal requirements extraction.

1.6.2 Governance analysis method

The method is an instrumental part of the framework and constitutes the main idea of this dissertation. At an abstract level, the method suggests the extraction and representation of requirements using a logic-based language for its eventual analysers. The method needs to be independent of the language of extraction and the language of implementation. It will assume a first order logic approach in the representation.

The method is based on our core belief that states:

A legal requirements validation problem can often be represented as first order logic problem. In addition, formal analysis techniques -when applied to the logic model- are able to assist in legal requirement validation question.

Figure 1.1 illustrates a scheme representing the core idea of the thesis. The left side represents the legal requirements validation problem, and the right side represents the first order logic analysis problem. Our belief is that it is possible for a semi-automated method based on a requirements meta model to partially transform a problem from a legal requirements validation space into a first order logic analysis problem.

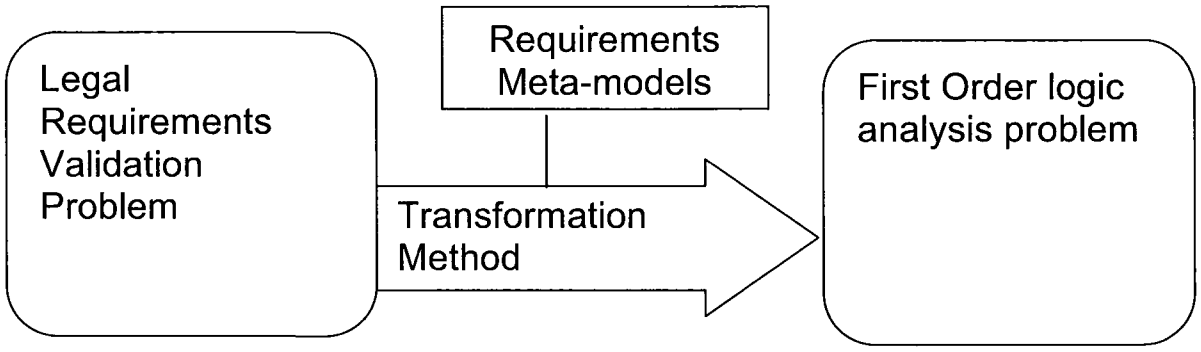


Figure 1.1: Core idea

The analysis performed at the logic analyser level can help us find violations or contradictions with possible examples given.

1.6.3 Governance Analysis Language and Tool

Language

The Governance Analysis Language (GAL) is a set of statements that can be used for defining governance requirements. Our language must be able to represent enterprise and legal requirements and have the ability to express the enterprise ontology and scenarios. It also needs to be adaptable to any changes in the meta model and be easily mapped into the logic analyser language.

Tool

The Governance Analysis Tool (GAT) will implement our method using the semantics of the meta-model, in addition to having a graphical user interface. The GAT should be able to import the requirement representations written in the format of our Governance Analysis Language (GAL). It should also be able

to produce a formal model that can be checked when using a logic analyser. The analyser must check for consistency and produce instances showing possible violations.

Usage

This process has to be executed by the GO and begins with a manual requirement extraction in addition to a formal representation. The second step is the automated translation from the requirements to a logic analyser model ready for analysis. Then, a consistency check using an analyser tool is initiated to ensure that the enterprise and legal requirements are not in conflict. The user interface will provide a scenario diagnosis that can be interpreted by the GO.

1.7.0 Thesis outline

Chapter 1 introduces the thesis and provides a hypothesis, defines challenges and proposes contributions.

Chapter 2 provides the underlying principles required for the thorough execution of this thesis.

Chapter 3 presents the background and the related work.

Chapter 4 presents the principal method and the tool we propose.

Chapter 5 presents the extraction patterns.

Chapter 6 shows the GAT Translation.

Chapter 7 illustrates an implementation of the method in the privacy domain.

Chapter 8 extends the implementation to the financial domain.

Chapter 9 provides details about the language used Alloy, the Tools used, and the analysis of the method.

Chapter 10 offers concluding remarks.

Chapter 2

Framework Principles

2.0 Introduction

In Chapter 1, we introduced our framework's main components, the method that describes our framework's process, the meta model that defines the framework semantics and the tool that provides the implementation. In this chapter, we will discuss some detailed aspects of each component. First, to elaborate the method's definition, we will discuss various compliance validation goals and methods. Second, to further explain the proposed meta-model, we will describe the enterprise and its legal models. To provide more detail on the possible deployment architectures for the proposed implementation, we will show a potential deployment diagram in section 2.7. Under the compliance component, we will describe the method. Finally, for the implementation aspect, we will show a potential deployment diagram for the tool.

2.1 Validation methods

As previously mentioned, there are three methods we will employ in this thesis to validate consistency and completeness: Model consistency, ontology, and scenario checking.

2.1.1 Model consistency

One of the important methods for compliance validation is to check the model for logical consistency or for the lack of logical contradictions. A logical

contradiction consists of a logical incompatibility between two or more propositions. It occurs when the propositions taken together yield two conclusions, which form the logical inversions of each other [Priest 2006]. For example, a contradiction can occur between a user-specific right and a group policy affecting the same user. At inception, it may be possible to detect if a group policy is violated by an existing right. In the case of a violation, it should be asked whether the user-specific right should be upheld as an exception. Furthermore, an enterprise may wish to check whether an enterprise policy as a whole is consistent with applicable laws. Model consistency checks are neither simple nor efficient to program, in the case of propositional logic, the required algorithm is NP-complete [Floridi 2007] [Fagin 2003]. However, some available software packages are able to find inconsistencies efficiently within bounded scopes such as in Alloy[Jackson 2006] .

2.1.2 Ontology checks

Another method we propose for compliance validation is to perform ontology checks. The governance laws specify requirements for the enterprise ontology, if an organisation's ontology does not match legal requirements its compliance is at risk. An ontology check ensures satisfaction of legal requirements. For example, the law may say that an enterprise must have a policy-audit department. An enterprise ontology includes elements such as the enterprise department and role structure in addition to process structure and related activities. Gaps in enterprise specification are considered legal violations as

far as the privacy law is concerned. Hence identification of such gaps is needed. Enterprise requirements may include multiple ontologies. However, for the purpose of this thesis, we will direct our attention at the organisational structure ontology and the process ontology. Our enterprise meta model does not have a legal ontology. The study of such an ontology will be left for exploration in future work. Ontology specifications are not restricted to e-businesses, traditional regulations also specify ontology requirements. For example, a city could have an ontology specifying the streets in the downtown area. It could then have a regulation saying that parking on downtown streets at night should be fined. Successive regulations could impose different fines for different street names. However, Murray Street, which the ontology specifies to be a downtown street, might not be mentioned in these regulations. This would be an example of incompleteness in the regulations.

2.1.3 Scenario checks

Another important method for compliance validation is to perform scenario checks. A scenario check validates that the enterprise system respects scenarios, which are required by law. In addition, it validates that the exclusion of certain scenarios from the enterprise process. A scenario, according to our method, is represented as a logical formula possibly containing structural, sequencing, and conditional propositions. A governance officer should have a tool available to simulate execution of certain scenarios and hence validate if they are compliant. Laws usually specify undesirable scenarios. A GO validates the in-existence of

these scenarios while checking for compliance. Conversely, laws have many desirable scenarios that need to be adhered to in order to declare an enterprise to be compliant. For example, a privacy policy of a company may specify that credit card information must be removed from the company's database after the transaction's purpose is achieved. It should be possible to test this case in order to validate that there is a process or an activity for the removal of this information. A single law may have many scenarios that need to be checked.

2.2 Enterprise compliance properties

In this section, we will define two enterprise requirements: legal consistency and completeness. To check these properties, we use three methods model, ontology, and scenario checking. Legal Consistency is a necessary property for a compliant enterprise. To check if the enterprise is legally consistent with the law, it is necessary for an enterprise to satisfy three checks: The enterprise requirements model satisfies model consistency checks; the enterprise requirements model contains no ontology violations; the enterprise model passes scenario violations checks (section 2.1). For example, the law may state that the audit department belongs to the governance division, whereas the corporation may have the audit department as a part of their financial division. Legal Completeness is the second necessary property for a compliant enterprise. Checking completeness aims is to ensure that the enterprise requirements permit all legally acceptable scenarios and deny unacceptable ones. As previously mentioned in Chapter 1, this fact can be checked by

ensuring that there are no scenarios possible under the law that are not possible under enterprise requirements. For example, the law may state that a client should be able to request deletion of his data. If enterprise requirements are unable to support such a scenario, they are said to be incomplete. Furthermore, a check to ensure that all possible situations are covered can also be implemented; we refer to such a check as a coverage check. We will leave the legal completeness definition and detailed examples for future work.

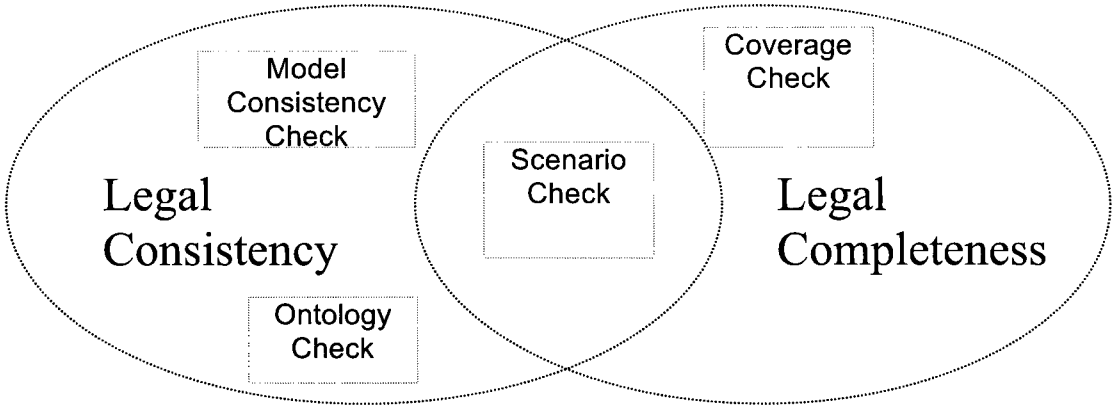


Figure 2.1: Methods applied to check for legal consistency and completeness

Figure 2.1 shows the proposed methods for consistency and completeness checking. The square boxes represent the methods, which we have partially presented in the previous section: Model consistency check, scenario check, Ontology check, and coverage check.

Legal Compliance: A set of enterprise requirements is considered compliant with the law if the requirements are legally consistent and compliant with respect to the law.

2.3 Governance requirements representation

As mentioned in Chapter 1, governance requirements can come from enterprise and legal sources. Governance requirements are a specific kind of normative requirements, which we will further discuss in Chapter 3. This section discusses the abstract representation of governance requirements. It exposes the challenges and presents a proposed legal requirements model as well as some examples.

2.3.1 Requirement modelling (Challenges and Method)

Translation of legal norms is faced with difficulty of translating general and externally dependent provisions that exist within the legal text. The law includes general statements that are high-level and thus are not easily represented in logical terms. Furthermore, these provisions are intended to hold various interpretations depending on the context of the application [Podgórecki 1974]. Dependencies do exist between different laws, details are also interpreted by the judiciary [Benditt 1978]; this interpretation forms case law. Most importantly, laws are dependent on an enforcement mechanism through policing services [Bailey 1995].

Figure 2.2 shows an abstraction of the GAM extraction method. Based on our analysis of legal requirements we have created an extraction model. The model will be used to provide an abstraction for the extraction patterns.

The abstraction patterns should closely match requirements in the law. Hence, from a user perspective, the GO needs to match legal requirements to the extraction patterns. For more detail see Chapter 5.

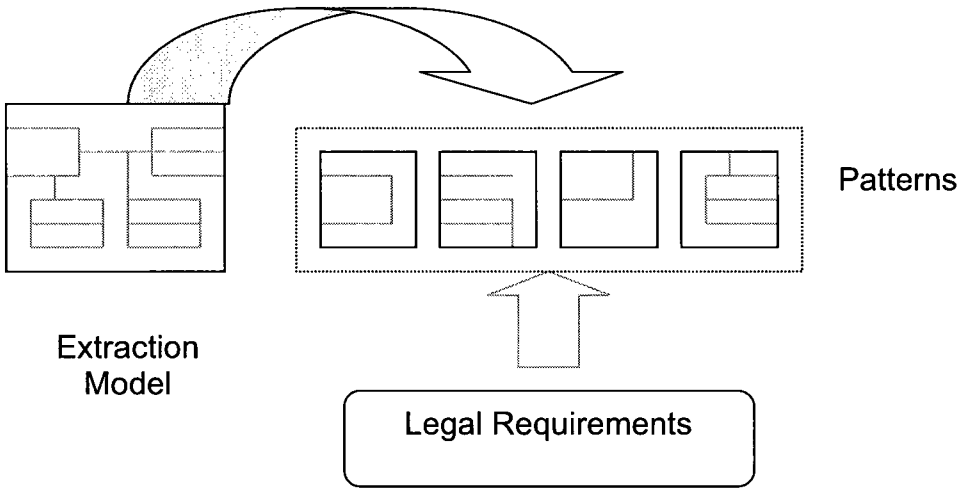


Figure 2.2: Legal requirements method

Consequently, the generality and the dependencies of the legal norms may make them difficult to interpret in precise terms. However, logical formalisms such as those necessary for the type of analysis we propose are designed to be precise and coherent.

2.3.2 Legal requirements model

Our method is dependent on a requirements model that provides a semi-formal representation of the entities and their relations. This model defines the semantics of our extraction method. The model is based on our analysis of the governance provisions taken from the privacy and financial laws in Canada and the USA. Figure 2.3 presents the class diagram of basic patterns. The model is also based on our analysis of legal script. It illustrates the requirements extraction model, we can see the process class with a potential relation to other classes through separation of concerns.

An activity may belong to a process and possibly be sequenced with other activities. It may also be assigned to a role, which has the possibility to be delegated to another role. The entity is composed of a set of roles and processes.

Sometimes activities are attributed to certain processes. For instance, a department can be composed within another department using the 'includes relation', whereas a process can be composed with another process using the 'composed of' relation.

Requirements may be directly implemented in the extraction model, but some refinement may also be required. A requirement may suggest a global scenario that needs to be validated or that a process may possibly be placed in separation of concerns. Furthermore, according to the law, roles can be delegated and may also have activities under their realm.

Once they are delegated, a role occupant or user gains access to the additional rights in accordance with the requirement.

See below the definitions of classes and their respective relations.

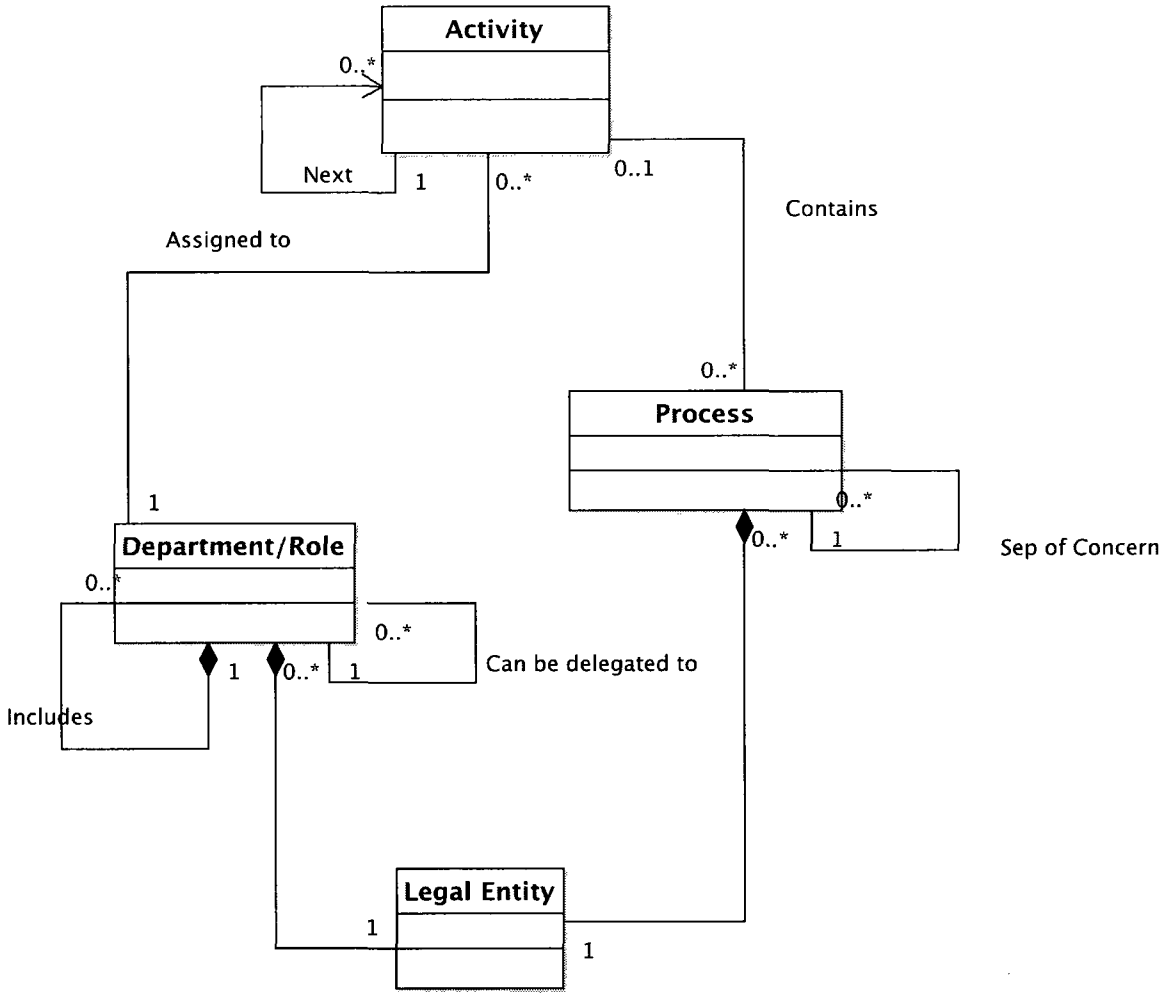


Figure 2.3: Requirements extraction model

2.3.3 Sample legal rules

In this section, we will provide a listing of some sample rules that dominate the governance laws. A discussion of the potential classifications and backgrounds is presented in Chapter 3.

Conditional statements: Requirements can be conditional statements. An example of a conditional statement is the "Consent Principle-3" of PIPEDA. It indicates that when an individual expresses a withdrawal of consent, the organisation needs to inform the individual of such implications.

Access-Right statements: Governance requirements may include access-rights such as: a project manager can be given access rights to project financial data. An example of a user assignment right might be User A assumes Loans process and is assigned to task Receive application. User B is assigned to process credit-check.

Separation of Concerns: Laws may also specify requirements for the separation of concerns. For instance: No data sharing between marketing and customer service; or no member of the governance board to be a consultant.

Delegation of Authority Rights: An example of delegation of authority right could be the possibility for a role-R_1 to delegate his rights to another role-R_2. This statement indicates that User A is able to provide User B with some of his access rights.

Ontology requirements: Legal requirements contain multiple ontology definitions but we will only focus on two of these, namely, organizational structure ontology and process ontology.

Organisational structure ontology: An ontology requirement can specify a structural element, for example, the approve-credit organisational department is required. The law can also specify that approve-credit is called review-credit department. Furthermore, the legal requirement can specify that there are certain people attached to it. For instance, the PIPEDA states that the privacy officer role may be assumed by one or more individuals, ... and shall designate an individual or individuals who are accountable for the organization's compliance ... (PIPEDA). This statement specifies enterprise ontology structure requirements.

Process structure ontology: Process requirements describe obligations related to the sequencing of steps or the existence of processes. For example, the PIPEDA states that the purpose declaration should be specified at or before the time of collection. It is also important to note that certain laws may add the possibility of role delegation.

Furthermore, ontology requirements include dictionary definitions, which describe the meaning of a particular activity. For example, according to the PIPEDA, individuals can give consent either by completing and signing a form, using a check-off box or articulating consent orally when using a particular product or service.

Complex types of requirements: Statements that cannot be implemented directly in logic form without undergoing a rewriting process, including the one we propose in this thesis. Such a provision may declare an invariant, that is, a property that must remain true. For example, the "Accountability Principle-1" (PIPEDA) states that an organization is responsible for personal information. This invariant may be further refined in the light of other statements in the law. For instance, an organisation shall designate an individual or individuals who are accountable for the organization's compliance; these individuals should be assigned to the privacy audit process.

Logic Operators: Legal scripts include statements that can be translated to logic operators. For example, there exists a process for data-disposal. Logical operators can join, disjoin, or assure the existence or lack of a particular requirement.

Attempts to categorise normative styles was made by [Sartor 2005] [Lorgippo 2008]. Chapter 3 will include a detailed discussion on existing and potential classifications of legal and enterprise requirements.

2.4 Enterprise governance model

In this section, we will present some background information on enterprise governance model by explaining the important building blocks. We will also provide a core meta-model that can be customised through the provided patterns.

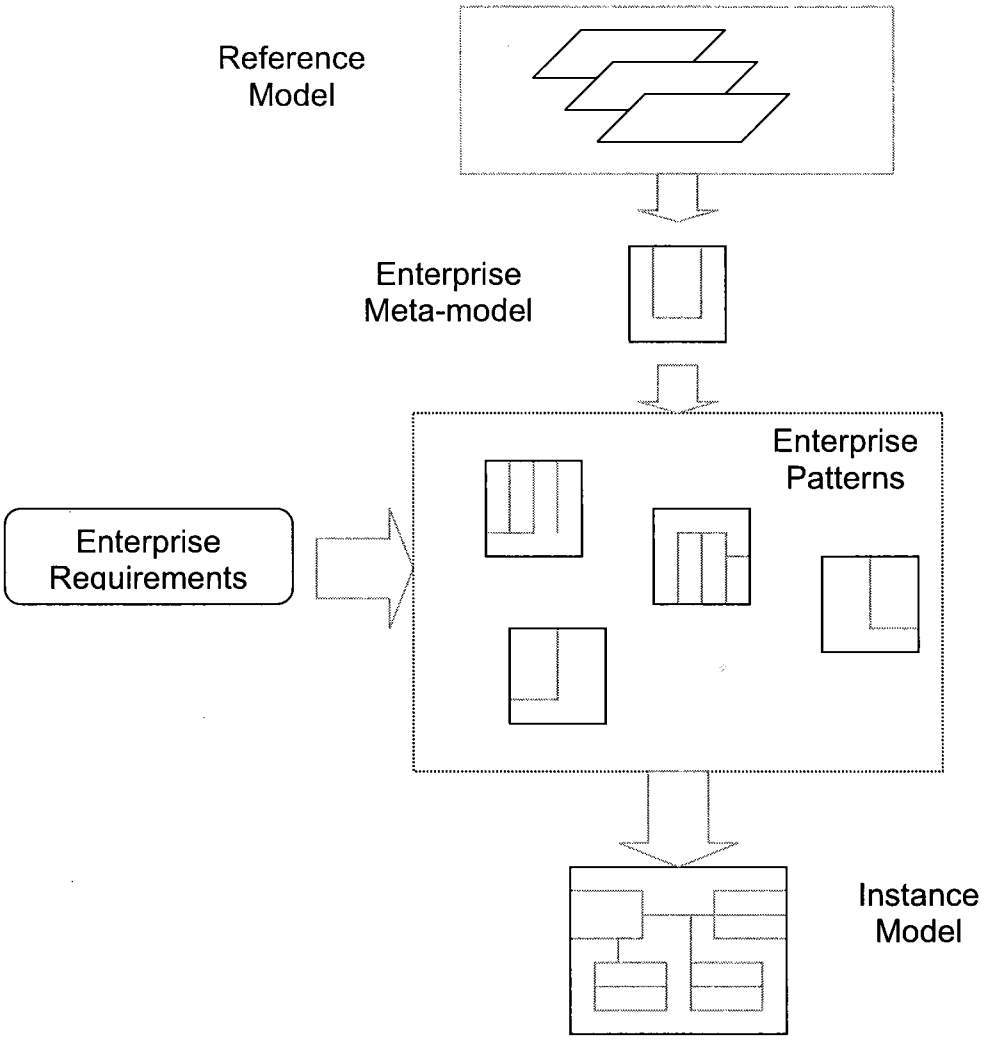


Figure 2.4: Generating governance model

Figure 2.4 shows our method for defining the enterprise governance model. First, we present a reference model, which provides a conceptual basis for our enterprise definition. Based on the reference model we propose an enterprise

meta model. The meta model can be instantiated by joining with enterprise patterns.

The resulting instance model reflects a given enterprise model. The details will be shown in the following subsections. Finally the enterprise requirements are mapped to the patterns as per the enterprise model instance, as shown in Figure 2.2.

2.4.1 Enterprise reference model

The enterprise reference model, shown in figure 2.5, is the basis we use to conceptualize enterprise elements. This model was presented in

Our view suggests three planes:

1. Subject and role-grouping plane: In this plane, the subjects are grouped into roles. Roles reflect subject access rights into the processes and activities of the middle plane.
2. Process and activity plane: Here, processes are put in a hierarchy and include activity graphs.
3. Object plane or data plane: This is the plane of data object identifiers. Objects enclose data

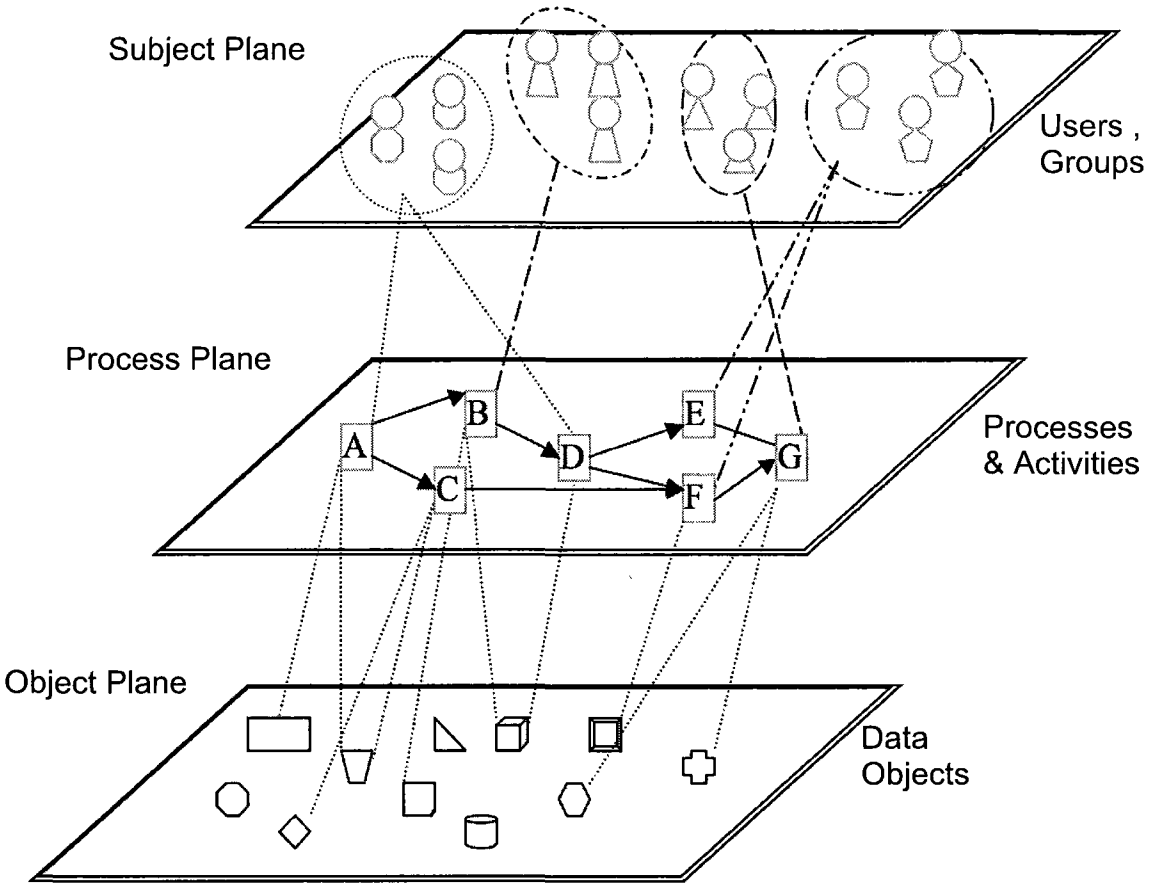


Figure 2.5: Enterprise reference model

These three planes are connected by mappings from the subject plane to the process plane. Mapping represents a logical association usually indicating right of access, or operating on an object to complete the process. Our method will focus on the top two layers of the reference model, namely the subject and the process layers.

The layers can be described as follows:

Subject plane

The *subject plane* includes the user groups and their roles. In enterprise governance requirements, a user or a group of users (a role) can be the subject of legal requirements. For example, the privacy or financial officer is a role defined by laws such as the PIPEDA and Sarbanes-Oxley (SOX). Role formations are not mandatory, but they are almost pervasive in enterprise definitions. There are numerous references in the legal requirements to role groupings.

Process plane

The *process plane* defines the process workflow. The process flow has the ability to implement process requirements, which are requirements that specify process compositions in addition to precedence relations between activities. The process plane acts as the intermediary between the subject and object planes. It assists in mapping processes to the object layer. A mapping defines an explicit 'reachability' relation from users to activities and to objects. Semantically, a relation between an activity and an object means that the activity has access to an object. Given that there is a strict mapping between objects and activities, we shall consider access to an activity equivalent to object access.

Object plane

The *object plane* consists of object references. These references can also refer to composite objects. Our method will focus on the top two layers of the reference model, namely, the subject and the process layers.

2.4.2 Enterprise meta model

The enterprise model defines the classes that are mandatory in any enterprise instance. Our method is based on its semi-formal representation. A meta model is a precise definition of the constructs and relations needed for defining an instance model. It establishes the boundaries and specifications of the instance models. A model is instantiated into classes and objects. Figure 2.6 shows our core meta model, including the business process class as a super-class.

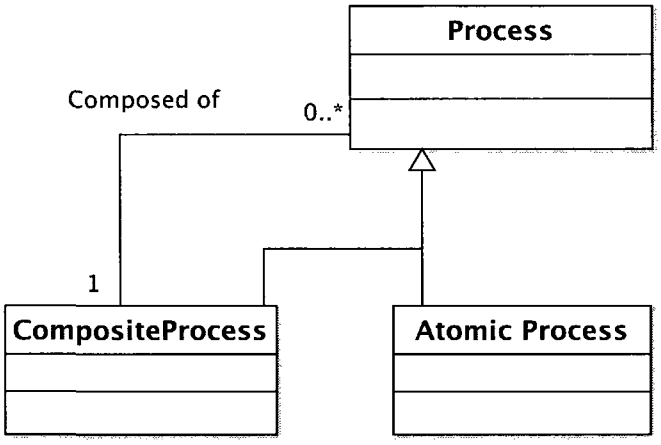


Figure 2.6: Enterprise UML Meta model

The Process super-class is composed as a UML generalization. The Composite process represents a super-class of the business processes. A composite process is composed of one or more processes. Atomic Processes are leaf processes solely composed of activities. Atomic Processes and composite Processes are subclasses of the business Process class.

2.4.3 Enterprise patterns

In this section, we will show a semi-formal representation of the enterprise patterns. A more precise and formal representation will follow in subsequent chapters. Because enterprises have various structures and their essential composition varies, we have defined several sample patterns to be used for instantiating an enterprise model.

Business process pattern

A process ontology is a tree of processes where each process either contains other processes or a sequence of activities, See Figure 2.7.

An atomic business process has a finite set of steps that form a *sequence.* A Process can be in separation of concerns with another process, (see Figure 2.8).

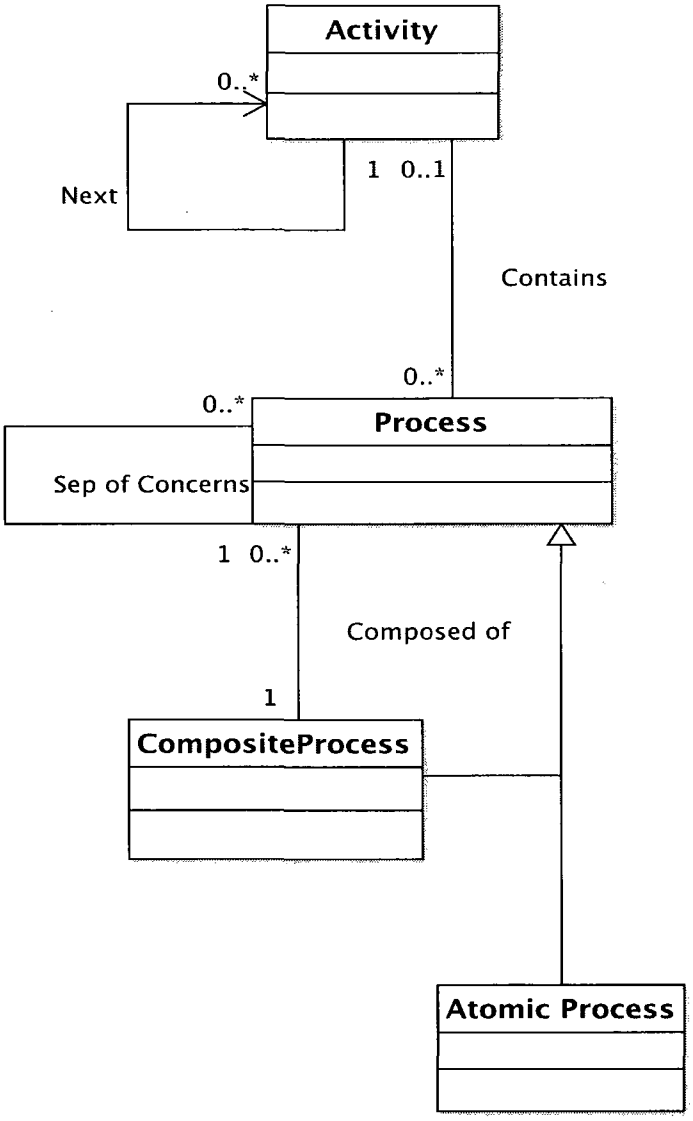


Figure 2.7: Business Process Pattern

Department/Role ontology pattern

A Department/Role ontology is a finite set of departments/roles with an ordered parent relationship, as illustrated in Figure 2.8.

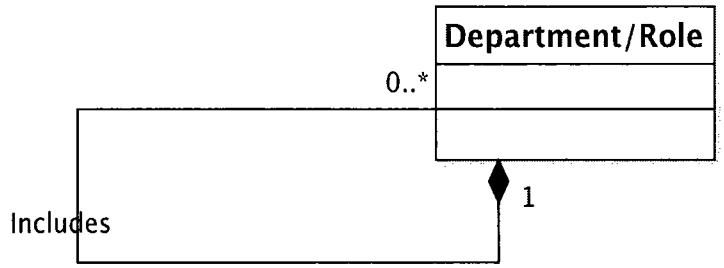


Figure 2.8: Department/Role Pattern

User/Role department pattern

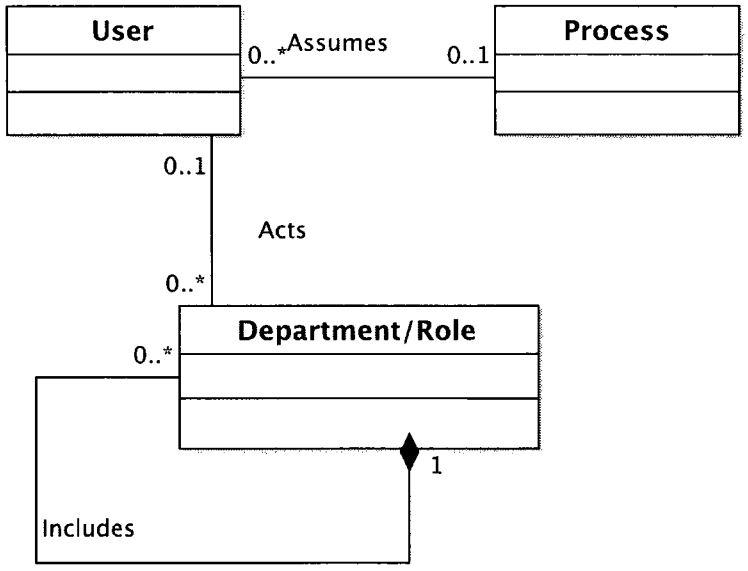


Figure 2.9: User Department/Role Pattern

Figure 2.9 shows that a user class consists of a finite set of users, where a user acts in some department/role in addition to assuming responsibility of some processes

2.5 Enterprise model

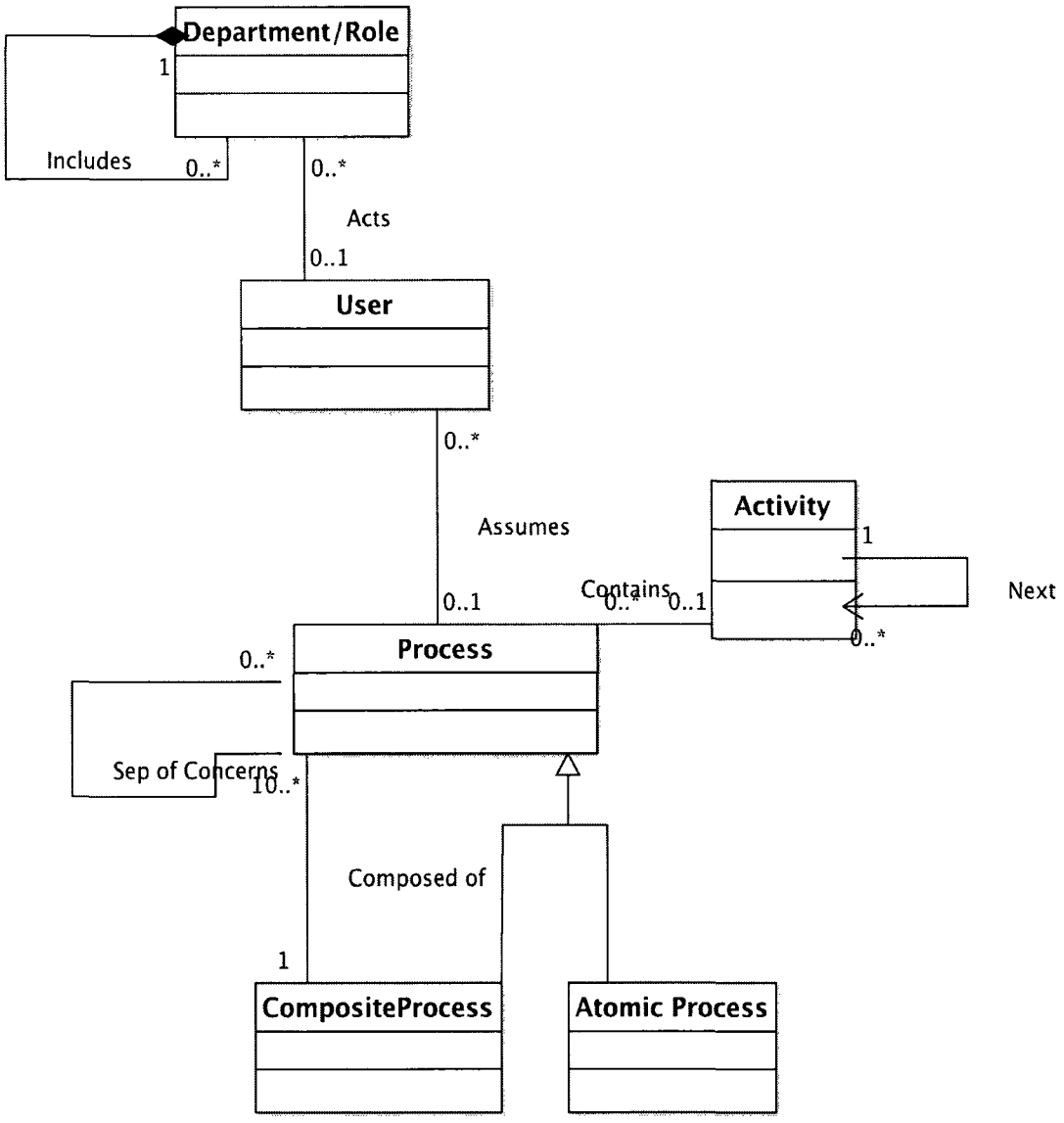


Figure 2.10 : Enterprise Instance Model

Figure 2.10 shows an instance model of enterprise and related relationships using a UML class diagram. It represents our view of how enterprise

requirements are defined. We represent the enterprise process using a graph where the processes are turned into a directed graph with possible loops and a definite termination. In other words, each process must have both a starting and ending step.

The *Process* class is a super-class. *Atomic Process* and *Composite Process* classes are its subclasses. *Processes* that are composed of one or more *Processes are instances of the Composite Process. Atomic Processes* are leaf processes that are composed solely of *Steps*. An *Activity* is connected to the next step or to no step at all. A *User* class can be assumed a *Process* through the *Assumes* relationship. A *User* can assume zero or more processes. A *User* can also act in a *Role/Department.*

A *Role/Department* includes none or several sub-departments.

2.6 Combined requirements model

In order to integrate requirements from the law and the enterprise we need an integrated model. The combined model needs to represent requirements from both sources.

Figure 2.11 shows our method of combining requirements. The enterprise requirements model, in addition to the legal requirements model will be combined to produce a combined requirements model. The rest of the thesis will refer to the model as being the combined requirements model.

Chapters 5 through 8, will use the combined requirements model. The extraction examples in these chapters will refer to it. The UML model will provide the meta-definitions of any possible instance. This combined UML model is illustrated in Figure 2.12.

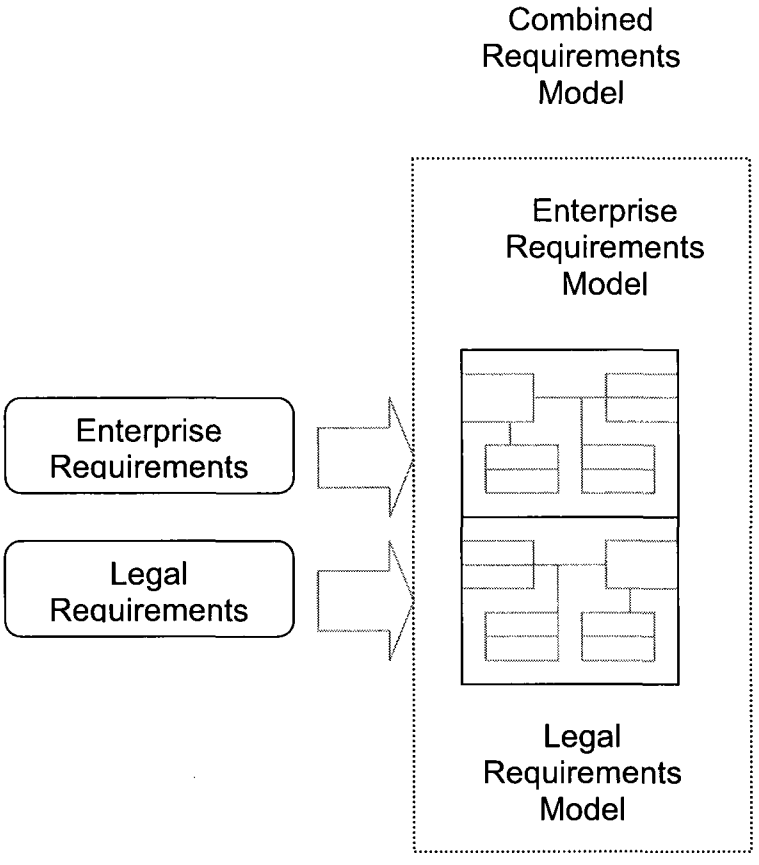


Figure 2.11: Methods process

We would like to stress the fact that our method intends to implement requirements using first order logic. The classes in the UML model will be represented using sets. The relations will also be presented either in sets or in logic formulas. Details of the logic presentation will be given in Chapter 4.

Starting from left to right and top to bottom, we describe the Department/Role class.

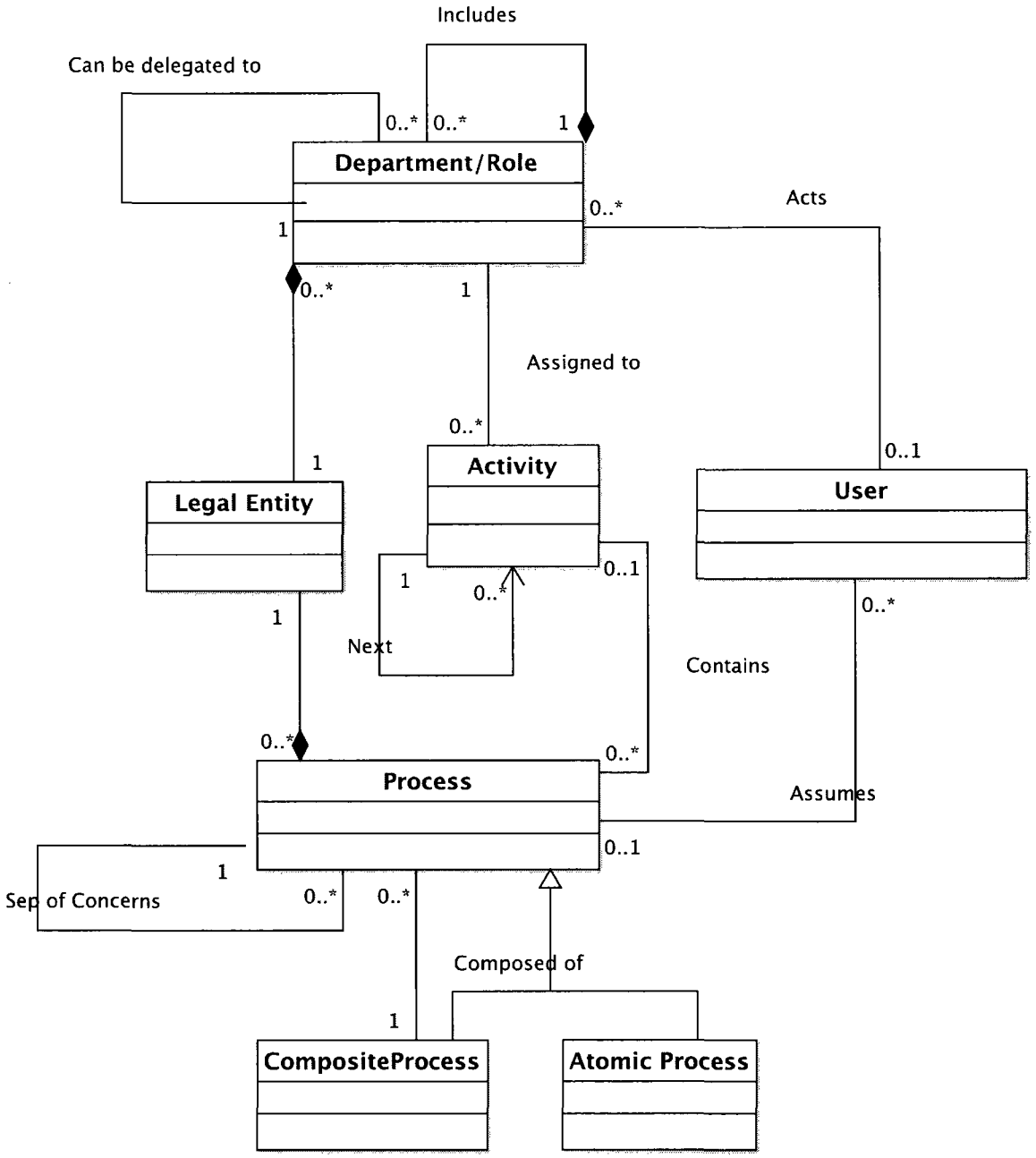


Figure 2.12: Combined Requirements Model

The Department/Role class instances are often explicitly stated in laws. For example, there are references to the financial or the privacy divisions. In such cases, the law requires the existence of these departments.

A role is usually assigned to certain activities in the law, such as *signing financial statements.*

Laws may specify role delegation rights or which roles are allowed or prevented from delegating their activities to other roles.

We have included a legal entity in the combined model. However, since this thesis does not study inter-organisational validation, this class will be ignored in further discussions. For reference, the legal entity class represents a named legal entity. Each legal entity is defined as a composition of departments and processes.

The Activity class refers to activities within processes. Activities are sequenced using the next relation. Atomic processes are composed of a chain of activities starting with a single initiating activity.

The user class is related to the role class, since a user acts in certain roles. Users also assume processes. When this happens, users receive access to all of process activities.

A process is either composite or atomic. The difference is that a composite process contains other processes while an atomic process contains activities.

Process can be bound to separation of concerns using the separation of concerns relation, which means that no users can have access to both processes at the same time.

2.7 Deployment environment

Presenting the context of an enterprise policy deployment architecture helps in setting the context for our proposed method and its future implementation. We base our discussion on the existing enterprise policy architectures, such as the one proposed by the RFC 2753 [Yavatkar 2000]. The IETF's RFC 2753, "A Framework for Policy-based Admission Control" has the goal of describing a framework for admission control. Our method can be implemented and deployed in accordance with the RFC recommendation. As our method validates the requirements; it also makes them ready for translation into policies.

The design of the RFC 2753 framework is based on five major components, See Figure 2.13. They are the Access Requestor (AR), Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Access Point (PAP) and the Policy Information Point (PIP).

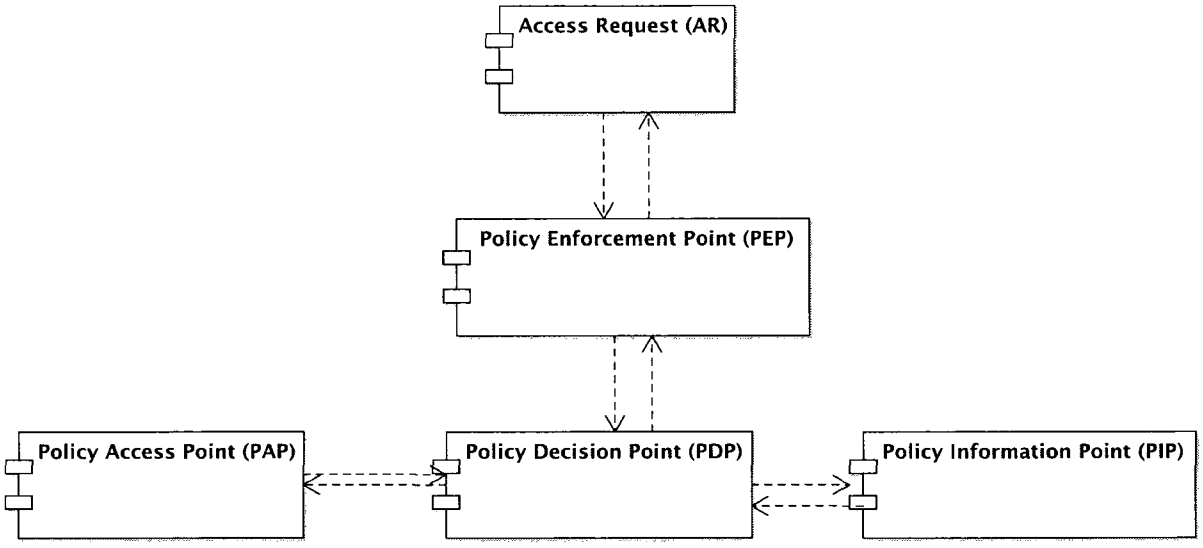


Figure 2.13: RFC2753 component model

1. The access requestor is any entity seeking access to resources.
2. The policy enforcement point enforces policy decisions, such as a firewall, router or the VPN system.
3. The policy access point is where policies are created and stored.
4. The policy decision point is where the decision is made, such as, the access control server.
5. The policy information point server is the source of attribute values or of requested data for policy evaluation.

Our method can, potentially, be attached to an architecture similar to the figure above. The design of a potential deployment framework, Figure 2.15, is based on five major components. They are the Requirements Access Point (RAP), the Requirements Validation Point (RVP), the Requirements Information Point

(RIP), the Requirements Compliance Point (RCP) and the Rule Translation Point.

1. The requirements access point is where the requirements are stored.
2. The requirements information point is where enterprise data, access rights and other requirements are stored.
3. The requirements validation point is where requirements are validated for inconsistencies after receiving input from the RAP and RIP.
4. The requirements compliance point is where the enterprise requirements are matched for compliance with the legal provisions.
5. The rule translation point is where requirements and specifications are transformed into rules or policies.

Our method produces verified requirements as specified in the RCP box. These requirements must be translated into policies currently not included in our method. The translated policies can then be deployed based on the RFC specifications.

2.8 Summary

In this chapter, we presented our framework and provided enterprise construction references and meta models. The enterprise models helped us define a possible enterprise instance. Furthermore, we discussed a deployment model for our method and tool and presented a combined enterprise and legal requirements model that will be used in the following chapters. The deployment model showed the possible architecture diagram for placing our proposed tool in a policy-aware enterprise.

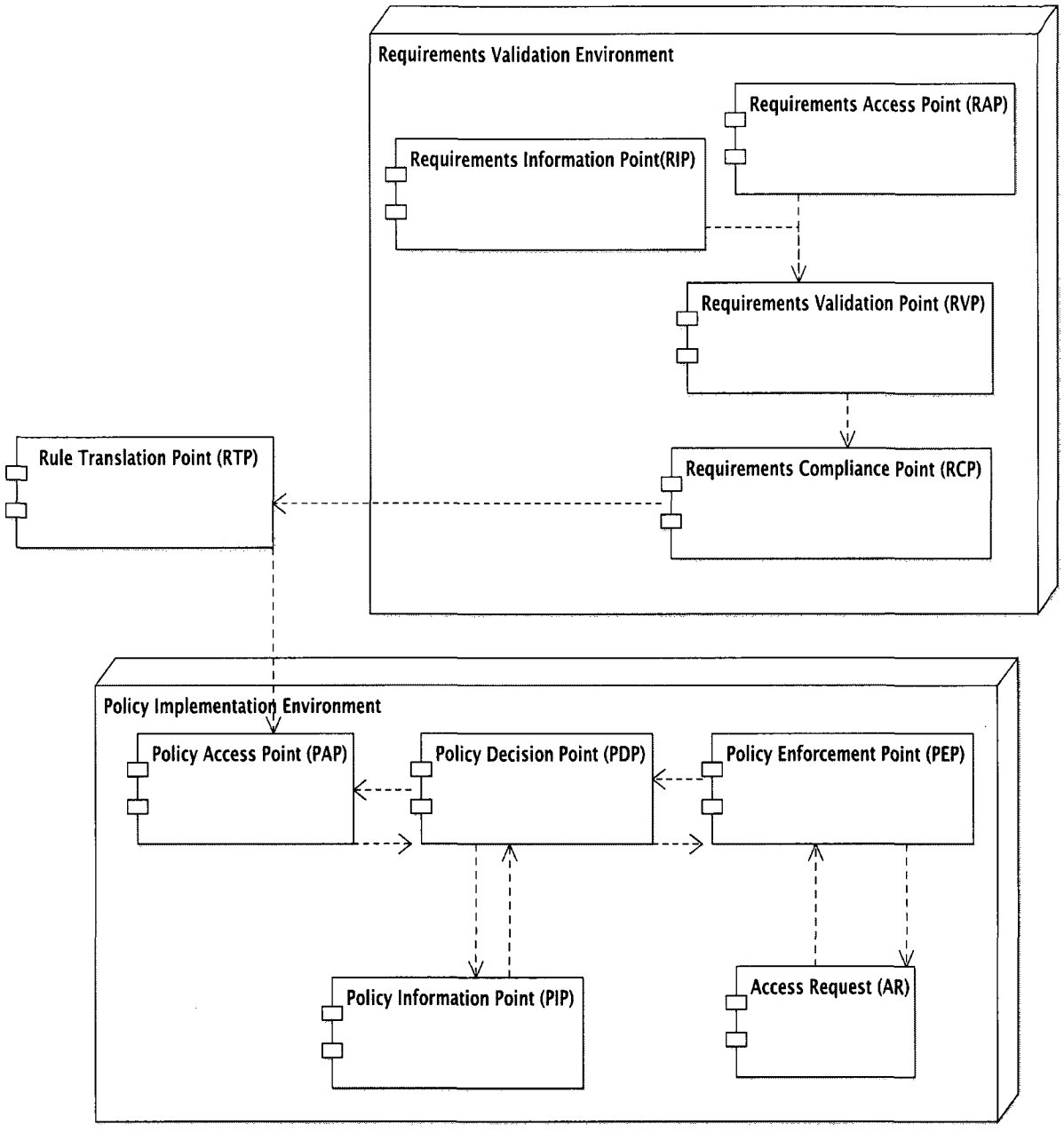


Figure 2.14: Potential deployment environment

Chapter 3

Background and related work

3.0 Introduction

This chapter discusses a high-level background of privacy and financial regulations and presents a formal analysis language in addition to related work on privacy and finance domains. We will discuss the different approaches to privacy and financial legislation, based on our study of North American and European Union laws. This chapter will also present challenges facing the process of extracting requirements from these laws. Finally, the chapter will provide a background on the existing research focusing on the extraction representation and the validation of the law. We will also highlight the legal compliance aspects that are not well emphasized in the existing research. The related work section will present a synopsis of other publications in the same domain.

Section 3.1 describes the state of affairs of the financial and privacy governance laws; section 3.2 presents the difficulty of extracting requirements from the laws; section 3.3 provides a background and discussion on the existing classification of the legal requirements; section 3.4 introduces the Alloy analyser language; section 3.5 presents related work including subsections on requirements extraction, implementation, and validation; and finally, section 3.6 concludes the chapter.

3.1 Governance laws

In the aftermath of high profile financial and privacy violations, corporate regulation has proliferated and scrutiny by the regulators, analysts and shareholders has intensified. As a result, corporations are driven to manage their compliance and governance obligations more proactively to ensure a timely and effective response to legislative change [GB 2004]. Legal entities, such as the enterprises and government agencies, must comply with governance laws. Governance laws are laws affecting corporate governance, including privacy and financial regulations and affecting public and private bodies [Dignam 2006] [Shleifer 1996] [Goergen 2007].

Table 3.1: Entities and regulations

Entity	Regulation	Domain	Notes
Canada	Instruments 52–109 and 52–111	Financial reporting	
Canada	Personal Information Protection and Electronic Documents Act – PIPEDA	Privacy	Commercial enterprises
E.U.	Guidelines for governance controls	Governance controls	Organisation for Economic Co-operation and Development
G10	Basel II Accords	Banking industry	
Global	IFRS	International Financial Reporting Standards	
ISO	Information technology – Security techniques – Code of practice – ISO 17799	Information Security Management	
U.K.	Turnbull Guidance and Combined Code	Financial Reporting	
U.S.	Health Insurance Portability and Accountability Act – HIPAA	Health and medical industries	
U.S.	Sarbanes-Oxley Act – SOX	Financial Reporting	Publicly-traded companies

Table 3.1 lists [Tarantino 2006] various governance regulations grouped by issuing entity and displays their regulation name, domain of discourse, and some related notes. As mentioned in Chapter 1, e-commerce companies are not only affected by their home country regulations, but also by those enacted in jurisdictions in which they do business or interact with partners. This explains why more e-commerce companies are working towards integrating requirements from multiple laws.

To mitigate the risk of non-compliance, companies may attempt to consolidate legal and enterprise requirements into a single corporate policy [Cannon 2006]. In this manner, coping with new compliance legislation is a simple matter of temporarily assigning a couple of people to determine the needs of the new legislation and updating the single policy where necessary [Cannon 2006]. However, this method is faced with many challenges, such as the changing legal ontologies. In a previous work [Hassan 2008-4], we discussed the importance of discovering applicable laws and representing their dependencies using ontologies as a main challenge for requirement integration. Second to integration, extracting the legal semantics and validating the model using logic is a challenge. Extraction is a difficult task due to the legal language complexity. Validation is a further challenge, because a well-defined model is required for systematic validation checks. For example, a common ontology combining the structure and relationship of the key enterprise classes from the

applicable laws is needed to understand the semantics of the enterprise attributes.

In the following two subsections, we will provide a brief introduction to the privacy and financial governance laws in Canada and the United States of America.

3.1.1 Privacy laws

Different countries have different approaches to legislating privacy [Bhatnagar 2004] [George 2003]. Approaches can be vertical, horizontal, centralized or global. Global laws encompass all industries, all government levels, and all activities. Vertical laws are industry specific, for example, healthcare, e-commerce or government.

The European Union, for example, enacted a Directive on Data Protection that relies on centralized and comprehensive privacy legislation. The United States' approach, on the other hand relies on industry-specific legislation. In Canada there is a hybrid approach to legislating privacy. There is a centralized privacy code, in addition to privacy laws at the provincial level (horizontal), most provinces have their own industry or government variants. The 'CSA Code' [CSA 1997] otherwise known as the "ten principles", serves as the centralized privacy law. PIPEDA the case study of discussion for this paper is a federal law for commercial activity. Ontario for example, a province, has a health specific

privacy laws PHIPA in addition to a municipal privacy laws for regulating municipalities [Hassan 2008.1].

Canada's Personal Information Protection and Electronic Documents Act PIPEDA is a vertical privacy law concerned with enterprises engaged in commercial activity. In the province of Ontario, the Municipal Freedom of Information and Privacy Act MFIPPA regulate privacy at the municipal level. Another example is the PHIPPA or Personal Health Information Protection Act, which is a privacy law for the health sector in Ontario. The PHIPPA is a combination of vertical and horizontal law, because it is specific to health at the provincial level. In Figure 3.1, we show a legal applicability map illustrating the vertical and horizontal laws and their application to various organisations. We can see the 'applies' relationship connecting the legal entities to the laws. Moreover, the dotted lines show the classification of the legal organisations in Ontario.

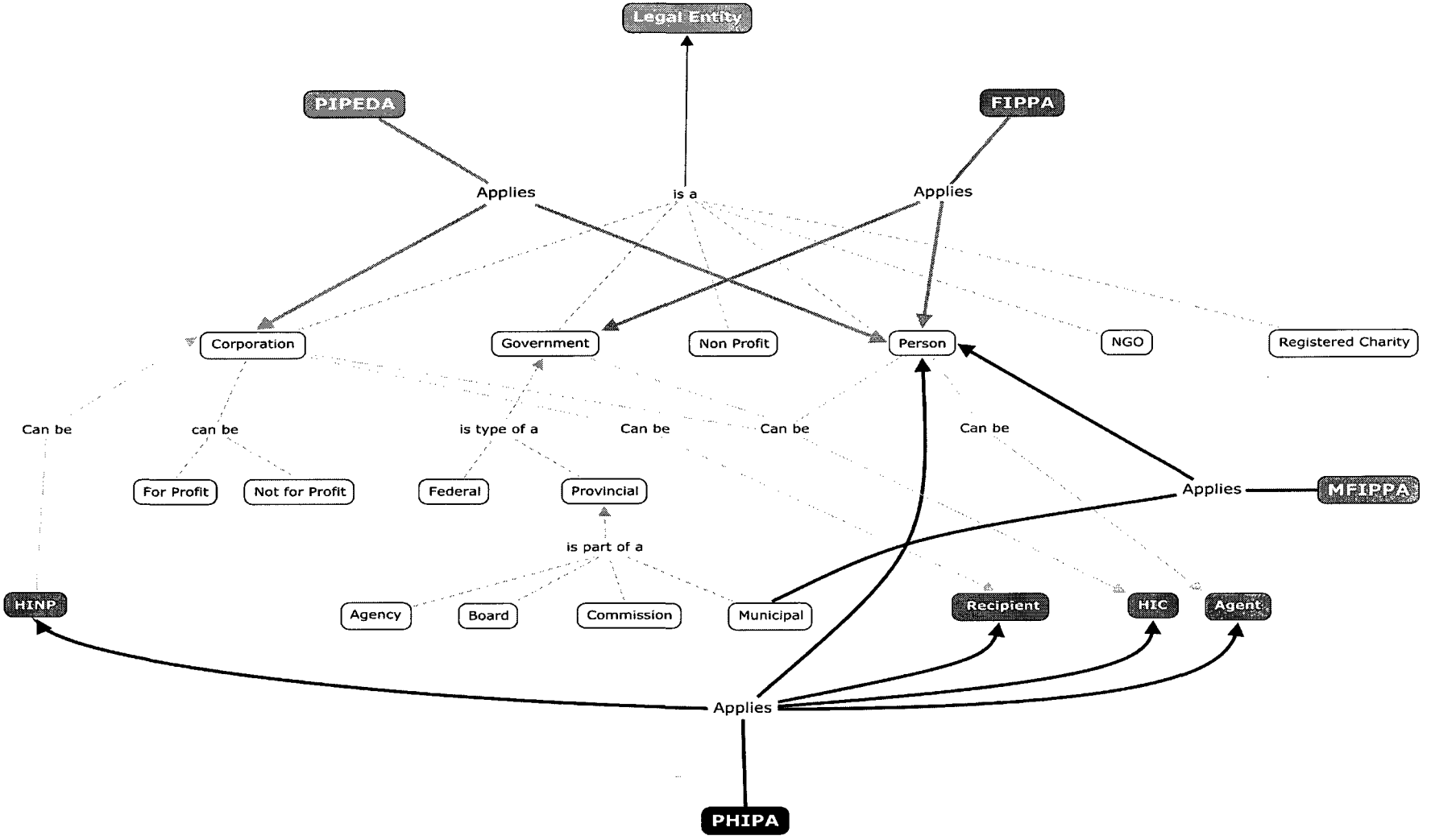


Figure 3.1: Legal applicability structure in Ontario

Table 3.2: Privacy acronyms used in Ontario

PIPEDA	Privacy Law for companies engaged in commercial activity
FOIPPA	Freedom of Information and Protection of Privacy Act
MFIPPA	Privacy law governing the operations of Ontario municipalities
PHIPPA	Privacy law governing management of health information in Ontario
HINP	Health information network provider
HIC	Health information custodians

In this thesis, we will focus our privacy discussion to the PIPEDA. This decision does not limit the scope of the principles and method described in this dissertation, as there are many privacy laws in Canada and abroad which are similar to the PIPEDA. Throughout this thesis, the PIPEDA may be referred to as the Privacy Act or simple the Act.

The Privacy Act requires organisations to obtain consent when they collect, use or disclose personal information; supply an individual with a product or a service even if they refuse consent for the collection; use or disclose personal information unless that information is essential to the transaction, and collect information by fair and lawful means; and have personal information policies that are clear, understandable and readily available. Though the Privacy Act requires affected organisations to comply with the CSA Model Code for the

Protection of Personal Information, there are a number of exceptions to the code where information can be collected, used and disclosed without the consent of the individual. Examples of this situation can be made for investigations relating to law enforcement or in the event of an emergency. There are also exceptions to the general rule, that an individual shall be given access to his or her personal information.

According to the main provisions of the Privacy Act, the implementation of the regulations is the responsibility of the corporations, which are subject to a commissionaire's review in case of subject (citizen) complaints. The scope of such an investigation will extend to the entities involved in the breach. Meeting regulatory requirements means that a corporation has to prove compliance. The implementation of multiple laws and directives suggests a complexity in the implementation and audit of privacy policy.

With respect to the generality of many of the PIPEDA's provisions, we have found that the number of specific rules, conditionals or declaratives is rather limited. The Privacy Act defines a data dictionary in addition to the privacy management process. Another interesting detail is that the law does not provide enforcement mechanisms. Instead, it defines the overarching processes involved. Although privacy complaints must go to the Privacy Commissioners' office, the enforcement and interpretation of the legislation belongs to the federal courts.

To automate compliance analysis, we must translate the law into a precise model with exact syntax and meaningful semantics. In this dissertation, we

take a formal approach by which laws are composed of logical and structural assertions.

3.1.2 Financial laws

Our discussion of financial laws is based on the U.S.A. law for financial reporting called Sarbanes-Oxley (SOX). We also bring examples from Canada's financial Instruments 52–109 [INST 52-109] and 52–111 [INST 52-111].

The purpose of Sarbanes Oxley (SOX) is both to enforce accurate financial reporting and to drive the development of stronger internal processes and controls within those organisations for the creation and management of financial records and approval procedures [Braganza 2007]. SOX attempts to improve principles and accountability in the operations of the companies in the U.S.A. It has been considered a major comprehensive legislation in recent years in the U.S.A. business security affairs. SOX is administered by the Security and

Exchange Commission (SEC) and has 11 titles, namely the Public Company Accounting Oversight Board (PCAOB), Corporate Responsibility, Enhanced Financial Disclosures, Analyst Conflicts of Interest, Commission Resources and Authority, Studies and Reports, Corporate and Criminal Fraud Accountability, White Collar Crime Penalty Enhancement, Corporate Tax Returns and Corporate Fraud Accountability. Two important points [Tipton 2006] of SOX influence the management of company records. The first point restricts the destruction, alteration, and falsification of records or documents. If an

individual attempts any of these activities, he will face severe penalties and potential imprisonment. The second point indicates that the businesses must follow a set of guidelines concerning communications recording, audits, records etc.

SOX also defines many retention rules, such as how long the business records are to be stored. In another example, it states that all business records including electronic records and electronic messages must be saved for not less than five years. Section 404 of SOX focuses on the critical role of internal control over financial reporting, emphasizing the importance of ethical conduct and reliable information in the preparation of financial information reported to investors.

Section 404 requires management and external auditors to report on the adequacy of the company's internal control over financial reporting. This is the most costly aspect of the legislation for the companies to implement, because documenting and testing important financial manual and automated controls requires enormous effort.

Under Section 404 of the Financial Compliance Act, management is required to produce an Internal Control Report as part of each annual Exchange Act Report. The report must affirm *the responsibility of the management for establishing and maintaining an adequate internal control structure and procedures for financial reporting*. The report must also *contain an assessment, as of the end of the most recent fiscal year of the company, of the effectiveness of the internal control structure and procedures of the issuer for financial reporting*. Both

management and external auditors are responsible for performing their assessment.

The financial reporting processes of many companies depend to some extent on their IT systems. Therefore, Information Technology controls that specifically target financial risks may be within the scope of a 'SOX 404 assessment'. Chief information officers are typically responsible for the IT organisation and IT personnel may be directly involved in SOX compliance efforts.

The SOX 404 guidance requires the use of an internal control framework. The SEC guidance states that ... *management only needs to evaluate those internal control rules that are necessary for the proper and consistent operation of other controls designed to adequately address financial reporting risks.* Some of these provisions state that:

Enterprises have an enterprise wide security policy; Enterprises have enterprise-wide classification of data for security, risk and business impacts; Enterprises have security related standards and procedures; Enterprises have formal security based documentation, auditing, and testing in place; Enterprise enforce separation of duties and Enterprises have policies and procedures in place for Change Management, Help Desk, Service Requests, and changes to applications, policies, and procedures.

Canadian Instrument 52–111:

In 2005, the CSA published for comment the Proposed Multilateral "Instrument and Companion Policy" ("CP 52–111"). The proposal CP 52–111 included Reporting on Internal Control over Financial Reporting, which would apply across Canada except in British Columbia. The effect of the new rule would be to ensure that Canada's reporting and certification requirements closely follow those of SOX.

CP 52–111 clarifies the scope of the evaluation because it relates to "underlying entities" including subsidiaries and other entities where an issuer has a significant equity interest but neither controls nor significantly influences the entity. The issuer is to take the underlying entities into account in evaluating the effectiveness of the internal controls over financial reporting. Management's evaluation of the internal control over the financial reporting must be based on a "suitable control framework". CP 52–111 refers to three suitable frameworks:

1. the Risk Management and Governance/Guidance of the criteria of the Control Board, published by The Canadian Institute of Chartered Accountants;
2. the Internal Control-Integrated Framework, published by The Committee of Sponsoring Organization of the Treadway Commission; and
3. the Turnbull Report, published by The Institute of Chartered Accountants in England and Wales.

These are a few segments related to 52–111:

1. 52–108: support the work of the new Canadian Public Accountability Board in its oversight of the audit profession
2. 52–109: CEOs and CFOs must certify accuracy of reports
3. 52–110: role and composition of audit committee
4. 58–101: information regarding corporate governance practices ("comply or explain" approach).

3.2 Difficulty of extracting requirements from laws

There is a growing interest in modeling regulations and managing them at an enterprise-level in compliance management systems [Wintgnes 2007] . Several examples are listed in Kerrigan, S. and Law [Kerrigan 2003] and Giblin et. al [Giblin 2005]. The growing demand for regulatory compliance for complex IT systems is facing the challenge of requirement extraction [Li 2008]. Governance officers need methods and tools to assist them in the analysis of regulatory documents and validation of their requirements.

We will now present the reasons for the difficulty of interpreting laws.

3.2.1 Difficulty of interpreting legal text

Legal text is written in a natural language in its full complexity [Bourcier 2003]. This makes it difficult to interpret and elicit the organisational requirements precisely, because legal documents can contain vague terms [Graham 2001],

complex dependencies between provisions, and legal lacunae. The complex dependencies between provisions make it difficult for an organisation to understand the full set of requirements.

3.2.2 Legal ontology

Organisations need to identify relevant regulations and extract legal requirements. A governance officer needs to discover the legal ontology to understand the structure and relationship of regulatory documents. This will help in understanding the relationships between the various information fragments. A lexicon ontology may also help harmonize semantics across multiple laws in addition to within the same law. Therefore, organisations shall analyse all relevant regulation and prioritize to identify the legal requirements to be met. For a deep discussion of legal ontology, see Valente's work in [Valente 1999].

3.2.3 Choice of extraction method

Legal texts are written in a natural language; thus, their interpretation is dependent on the method of extraction. There are numerous ways to implement laws into an intermediary language; the accuracy of the extraction is dependent on the model used. It is difficult to have a method that is able to automate requirement extraction from the laws. For example, given that a law may have deontic expressions, it is possible that an extraction method may have to decide whether to interpret these expressions or to pass the difficulty of translation

into the implementation environment. By way of interpretation, deontic expressions can be transformed into if-then-else type statements. Alternatively, the implementation environment may have deontic constructs that are able to capture the expressions without translations. The choice of the intermediary language syntax and semantics is non-trivial. An intermediary language that is closer to a natural language is difficult to implement, whereas an intermediary language that is closer to implementation prompts a larger gap between the requirements and the implementation. This gap can be bridged by the use of patterns where possible. The gap may otherwise be closed by reliance on the manual work of the enterprise governance officer. The manual process is usually error-prone and not repeatable.

3.2.4 Repeatability

It is apparent that compliance validation is incremental and ongoing and will need to resolve evolving standards, including new global regulations [Nettleton 2004]. Laws evolve continuously [Gavit 1952], and although it may be true that change takes time, the reversal of the verdicts and the changes in the regulations are, nevertheless, frequent. If we take the financial sector laws as an example, we can very well observe that they have changed recently owing to financial crises [Soros 2008]. In addition, countries are continuously revising their trade agreements, often adding new partnerships and updating existing ones. Trade agreements often change the legal landscape, because they affect the commercial companies' regulations. This is why a requirements extraction

method needs to be repeatable, without ignoring the fact that it is challenging to construct a method capable of adapting to changing laws.

3.3 Background on legal requirements

In this section, we will provide a background on normative systems [Alchourròn 1971]. We will also discuss the relation between normative systems and logic. Furthermore, we will provide a background on some existing classifications on normative styles. Finally, we will discuss the requirements domain.

3.3.1 Normative systems

Normative systems have been defined as sets of constraints on the behavior of agents in a system [Agotnes 2007]. Some examples of normative systems include corporate governance, legal and access-control systems, firewalls, and business application logic. There have been many attempts to translate normative systems into logic [Boella 2006]. If these methods are applied to laws, they would possibly improve the certainty of the law in addition to the ability to validate decisions [Jones 1994] . In relation to logic, we find that laws have the capability of expression in addition to specifying reasoning, branching and decisions [Sartor 2005]. Any formal analysis of legal norms requires a translation of the law into a formal representation. We believe that such a translation could be less complicated in certain legal domains than others. For example, criminal and family laws are highly dependent on social constructs and norms. Both are unstructured, thus making the translation of criminal and

family laws a difficult task, while at the same time holding little benefit. In contrast, the enterprise world is more rigid and structured. E- business laws and models have forced enterprises to structure and define their business processes and relations, internally and externally. Given the semi-static and heavily structured enterprise definitions, we see the enterprise domain to be more suited for the use of logic to represent legal requirements.

3.3.2 Legal styles

There have been several attempts at classifying normative styles in general and legal norms. It has been proposed that norms can be expressed in different styles of which two are particularly apparent, namely, the rule style and the requirements style [Logrippo 2007]. Normally, both styles are not explicitly distinguished and may not be explicitly present, but they need to be made explicit for analysis.

In a previous work [Hassan 2008-9], we extended the classification to include a third style, namely the ontology style. The ontology style is orthogonal to the first two. Sartor [Sartor 2005] has a somewhat complementary classification discussing many types at the meta level. The legal style discussion provided is important, because we provide a pattern-based classification as previously proposed. We will present existing work in this section, but would like to mention that the classification of legal requirements will be better explained in Chapter 5.

Rule style:

Rule style norms can be conditional instructions. Under certain conditions, these instructions require that certain actions be taken. Rules can also be procedural instructions. They are similar to Event-Condition-Action (ECA) rules [Chomicki 2003] or to Prolog programs [Sergot 1982]. An example of a norm in rule style is if a person does not pay the debt, the person's possessions will be sold. Another example is the PIPEDA "Consent Principle-3", which indicates that when an individual expresses a withdrawal of consent, the organization needs to inform the individual of the implications.

The level style specifies the final implementation of the law, which can be, by itself, capable of enforcing a legal system [Hassan 2008-9]. It has been proposed that it is possible to specify normative systems uniquely at the rule level [Hassan 2008-9]. Examples of normative systems written at this level are the Hammurabi code [Johns 1911], XACML policy systems [Ferraiolo 2007] [Periorellis 2007], and firewall rules [Lerner 2000] .

Requirements style:

Requirement level [Kelsen 1945] provisions are higher-order statements. They may express a desirable state of affairs; including obligations such as debts must be repaid. This style includes statements that provide legal power, that is, actions that affect legal decisions. It appears that this level cannot exist alone and that it depends on the rule level for enforcement and ultimate effectiveness.

In the PIPEDA, one finds a requirement style norm (Accountability Principle-1), which states that an organization is responsible for personal information.

A requirement level provision could possibly be written using one or more rule-level statements. For instance, requirement-level example may suggest that the silence of the subscriber for the renewal of his contract counts as an acceptance of modification and can be translated into a rule-style statement suggesting that if no response is received, then do not delete subscription. Of course, there may be other types of requirement-style rules; these will be presented in the discussion subsection.

Ontology style:

The ontology level is orthogonal [Hassan 2008-9] and expresses the domain structure usually common to both previous levels. To present the legal semantics precisely, we need ontologies that are often implicit in law. A very basic example is family law, because it cannot be understood without a reference to family ontology, that is, a definition of family relationships, such as child, spouse, parent etc. Similarly, to understand the laws relating to enterprise governance, we need to refer to enterprise ontologies. Here, enterprises are defined as hierarchical structures including departments or roles to which a process or steps can be assigned [Hassan 2009].

Laws can specify the relations between ontology elements. For example, financial controllers should report to the Chief Financial Officer (CFO) . The

ontology will define the two roles and their relationship in the organisation. A privacy law may specify that consent can be received through a signature, a check-off box or a verbal acknowledgement. This establishes an equivalence relationship between these activities.

More examples of normative style:

There are several other legal requirement styles presented by Sartor in [Sartor 2005], which we will not explore within the scope of this thesis. However, these types do exist within laws and need to be mentioned. They tend to be categorized under requirement level norms:

1. A requirement level provision can provide legal power. One example is if a client has the legal power over his personal information. Such a statement is generic and high-level. To see the effect of such a statement, we consider the following legal requirement: An enterprise has the right to retain information to achieve the purpose of the service; A person has the right to request a service; he also is able to revoke the organization's right to retain personal data. In a possible scenario, a person may request a service and hand-over his personal information. Shortly after, he can revoke that right even though the organisation may be under contract to deliver the service. In this situation, the user has exercised his legal power to supersede a conflicting requirement.

2. A normative proposition can discuss necessity or implied right. For example, a legal requirement may specify that all clients' private data-access must be bound to a purpose. Assume that a mail operator needs to access clients' addresses for shipping purposes. In this case, the enterprise policy should provide the mail operator with an implied right

based on necessity to deliver the purpose.

3. Another type of requirement level provision is one where actions declare certain legal states, hence declaring violations or compliance. For example, an organization can breach a person's privacy by forwarding his data to another organization without prior consent. This is another kind of high-level requirement where there is a condition to achieve a certain state. In this case, the condition is forward data without consent and the state is privacy is violated. The goal of the requirement is achieved, although it is not sufficient to determine the violation penalty. There may be other provisions stipulating that in detail.

4. Legal applicability norms may state laws corresponding to certain situations or may defer the decision to other laws. If personal information records include health information, the applicable provincial law applies. This is a hypothetical federal provision referring a specific scenario to a more specific law.

5. Priority Setting of laws is considered to be at the meta-level. Legal ontologies can be used to represent such applicability, and may prove useful for e-commerce transactions, given that they have multiple parties and may span across multiple legal jurisdictions. An ontology can also be used in a law conflict resolution in situations involving distinct vertical laws in the same jurisdiction, such as privacy and health laws in the provinces. They can also be used to validate horizontal conflicts at the municipal, state and federal (country) levels, and even NAFTA levels.

6. Finally, there are legal status modifying requirements, which are requirements with the tendency to change the legal status of their subject. For example, the signing of a collaboration agreement allows a recipient organization financial data access-rights over data collected by the donor organization. There are certainly more types of legal norms,

but they will not be further elaborated within the scope of this thesis. Their study will be left for future work.

Discussion

New styles in governance: We have found that legal requirements not only suggest ontology style statements, but also logic formulas operating over ontology statements. For example, an ontology requirement may mandate two roles to be specified, the CFO and the CPO, financial and privacy officers respectively. A rule may suggest separation of these roles accessing the same process, thus making the suggested rule that no one person can occupy both processes of privacy and CPO, hence creating a wall of separation. There is another type of context style rule that is called 'delegation of authority'. Under this style, the law may provide the ability of a particular role to delegate its rights to another role. A third type of these statements is the assignment sub-type, where a person can be assigned to a role or business process or may be denied it.

We have labeled these rules as being contextual, because they are appropriate to governance systems. Even though the concept of a legal delegate exists in the law, the governance laws provide a specific view.

Relation between rule style and requirement style: The relation between rule and requirement styles has been proposed by Logrippo [Logrippo 2007] . We conjecture that, in most cases, there seems to be a trivial translation between the rule style and the requirement style. For example, A is obligatory can be

translated to if a subject does not do A, then the subject will be punished. On the contrary, a rule can be generalized into a requirement. For example, if a partner violates the privacy policy, then notify the client can be translated into violations need to be reported to client.

A classical representation of the requirement style norms is by means of deontic statements. We conjecture that such requirements can be represented in rule level statements. For example, a client has the right to view his personal information can be translated into if the enterprise collects information, then allow the client to view it. Another example taken from financial law indicates that a client has the right to specify the list of competitors to prevent information theft. This can be represented by if a client specifies a list of competitors, then place the client in separation of the concerned group.

Conjecture:

We have noted that the categorisation made by Logrippo [Logrippo 2008] is generic and that it fits the normative systems in general. The work of Sartor is more specific to legal systems in due process, thus being one step further from providing a direct benefit for the automation of the legal systems. While we agree with the conjecture that normative systems are mostly composed of requirement and rule style provisions, we believe that governance laws are specific type of normative systems. Governance laws, such as the financial and privacy laws, provide new examples and types of requirements usually foreign

to social laws. We will now provide a background of rule and requirement styles and explain our recently proposed ontology and context styles.

3.4 Background on Requirement methods and notations

3.4.1 URN

User Requirement Notation (URN) was introduced as a standard by the International Telecommunication Union (ITU-T) in 2003. The main intention of this visual modeling tool is to help with functional (behavioral) and non-functional (e.g., availability, scalability, and cost) requirements. Since the defined objectives for URN are broad and ambitious, the following two components are used for achieving the desired purposes.

The first component is Goal Requirement Language (GRL), which combines the Non-Functional Requirements Framework (NFR) [Chung 2000] and i* framework [Yu 1995]. GRL's soft goals (that are shown as cloud symbols) allow depicting objectives with ambiguity about their level of satisfaction in the system. Soft goals can be decomposed and divided further into sub-goals to reach a quantifiable and operational solution. The operational part can be illustrated as tasks. One of the advantages of GRL over other modeling languages is its higher level of abstraction that helps us find out the opportunities and vulnerabilities [Yu 1997]. The second subset of URN is Use Case Map (UCM), which can be used for scenario definition. It is a useful notation to define behaviors of the system both in top level and operational level

processes. In other words, it is general enough to be used for defining a business model or to define low-level activities and responsibilities in one portion of an implemented system [Weiss 2005]. The tractability between UCM and GRL also allows us to find out the defined goals that are not covered by our operational system [Pourshahid 2007].

3.4.2 i* framework

The i* (distributed intentionality) framework proposes an agent-oriented approach to requirements engineering based on the intentional characteristics of agents [ISTAR 2009]. As mentioned, it incorporates NFR framework's softgoal and subgoal contributions for goal refinements [Chung 2000].

The framework is applied in contexts that comprise multiple parties with strategic interests that might be conflicting or synergistic. These include information systems requirements engineering, business process modelling and redesign, software process modelling [Yu 1997].

The i* framework process consists of (a) identifying the actors (b) goal/task identification and (c) dependency identification. The elements include [ISTA 2009] [Liu 2003] [Onabajo 2009]:

1. An actor - a unit, which can be ascribed intentional dependencies. This can be a role, an agent or a position. A role is an abstract actor with some responsibility (e.g., patient, administrator) while an agent is a concrete actor with particular capabilities and functions (e.g., John Doe, PDA device). A set of roles , as a package, can be assigned to an agent is called

a position.

2. Goal - a condition or state an actor would like to achieve. Goals could be softgoals, which are typically non-functional [56]. A hard goal refers to a function.

3. Task - a course of action to produce a desired effect. It represents a specific procedure to be performed by an agent.

4. Resource - a physical or information entity [56]. Agents attribute intentional properties (such as satisficing softgoals, achieving goals,

3.5 Related work

In this section, we will present related work. In addition, subsequent chapters may include further detailed discussions and correlations with existing work.

Our work falls into the category of legal compliance validation methods and tools. Very broadly, our approach is related to the methods used in the area of legislative drafting systems, telecommunication control systems, network security and firewalls.

The early approaches of representing legal requirements related to privacy, security and financial control included a variety of XML-based languages, often with partially-competing goals. These languages lacked the formal semantic models that were suitable for capturing legal requirements and definitely did not have the ability of formal analysis. P3P is a language for specifying privacy policies for web sites. XACML [Moses 2007] is a declarative access control policy

language implemented in XML whose semantics are described by a processing model in describing how to interpret the policies. The Enterprise Privacy Authorization Language (EPAL) is used for writing enterprise privacy policies to govern data-handling practices in IT systems according to fine- grained positive and negative authorization rights. These languages are intended to implement privacy requirements but do not have precisely defined semantics [Schunter 2007].

Formal compliance validation requires a rigorous method based on formally defined syntax and semantics in addition to formal validation. Several research papers including [Brodie 2006] have discussed regulatory compliance. Such an approach has been validated by some industry work recognizing the ability of computing systems in the compliance validation process [Kudo 2007] . This thesis presents a step towards providing the ability to formalize certain governance requirements and partially automate compliance analysis. Our approach is far from covering all aspects of enterprise governance laws; in fact, we are not trying to approach completeness, because ethical, social and other aspects can be impossible to represent in logic-based semantics. We shall show that some formal coverage is possible and that formal compliance analysis and formal auditing is also possible for those aspects we can express. We believe that the method and tool presented are novel with respect to the use of logic and the ontologies to validate compliance.

Our approach corresponds to Hambrick's work [Hambrick 2008] by laying out the importance of the relationship between enterprise components, such as formal structure and behavioural process on one hand, and a legal and normative compliance on the other hand. Our proposed framework also corresponds to the Antón et al. 'roadmap' that is in line with [Antón 2007]. We are in line with the approach suggested in [Cannon 2006], which places the first research point as the consolidation of the policy management and the second as automating compliance. In this dissertation, we plan to extend our work to generic corporate governance, hence consolidating financial and privacy policy compliance.

We will differentiate between three kinds of related work. The first is at the requirement extraction level, the second is at the requirements specification level and the third is at the implementation level. Finally, we reference various problem approaches.

3.5.1 Requirements extraction level

Most work on the semantic translation of requirements [Ashley 2004] [Balust 2001] [Fischer-Hubner 1998] [Li 2003] [Mitra 2006] [Roessler 2007] takes a data-centric approach. Such approaches focus on the extraction process on electing access-control provisions over data items. Some work on requirement representation based on the choreography of the services has been presented in [Amyot 2001]. Proponents of formal methods have claimed to resolve the

problem by providing unambiguous and mathematical notations and verification techniques [Jarke 1998].

Onabajo et al. [Onbajo 2009] propose CREE -Confidentiality Requirement Elicitation and Engineering, supports analysis of confidentiality requirements through stratified goal models. CREE analysis is complemented by goal modelling through semantic annotation of text from source documents. The annotated texts represent concepts, which are used in the creation of goal models and subsequent analysis. Traceability between goal model elements and the text is established through the annotations, therefore addressing one of the issues with requirement extraction from natural language regulations. Their work includes a structural analysis based on OCL constrains.

Authors in [Breaux 2006] recognize the role of constraints in identifying the conflicts between rights and obligations. They provide a process called 'semantic parameterization' to derive rights and obligations from privacy goals.

A methodology is presented in [Breaux 2008] for directly extracting access rights and obligations from regulation texts. Similarly, the approach in [Kiyavitskaya 2007] starts with the text analysis of the law by proposing a tool and methodology for extracting (rights and obligations) from the legal requirements. In contrast, our work does not cover the extraction issue, making us consider these methods as complementary.

An ontological approach in [Lee 2005] applies extraction of requirements from regulations. This approach categorizes the requirements by using an ontological model, thus helping to rigorously identify the inconsistencies between the model and the regulations. Our approach is also dependent on the ontologies for representing the enterprise specification. Other related research has been carried out on text extraction of policies from laws [Brodie 2006] [Kiyavitskaya 2007].

3.5.2 Requirement specification level

Sartor and Logrippo [Sartor 2005] [Logrippo 2007] have proposed logic abstractions of the legal concepts. [Antón 2007] suggests that at the specification level lies a need for the development of a formal language for specifying privacy in addition to an automatic translation mechanism from natural language privacy policies to formal language policies. Among other examples, at the enforcement and audit levels, there is a need for a theory for information flow control based on privacy policies. Their identified goals align with our research goals and strategies.

Enterprise requirements models have studied by us in previous work [Hassan 2005] [Hassan 2006-1] [Hassan 2006-4] [Hassan 2007] [Hassan 2008-1]. In [Hassan 2005] we proposed a process based enterprise model containing roles, process, and activities. In [Hassan 2006-4] [Hassan 2008-1] we provided an elementary model for enterprise requirements separating the enterprise into

three layers, subjects, verbs, and objects. [Hruby 2006] also contained a study of a model based on based on resources, events, and agents (REA); REA was originally proposed in [McCarthy 1982] as a generalized accounting model. We find that our reference model is granular. It provides fine-grained control over enterprise resources. The ability to represent flows is also suitable for matching legal requirements, which often specify process flows.

3.5.3 Implementation level

The use of formal methods for capturing the concepts of laws is proposed in [Otto 2007]. Their survey lists several techniques including the use of symbolic logic, implication, conjunction, etc., knowledge representation using PROLOG, deontic logic using LEGOL, defeasible logic, first-order temporal logic, direct access control, markup-based representations, and goal modeling. Although the survey includes symbolic logic, which we also use, it does not include graph models as a potential formalism. Other related work is presented under the banner of legal programming [Subirana 2006]. Our aim is to look at compliance issues and hence implementing laws as programs is not a concern of ours.

An important related work is by [Barth 2006]. Their work presents a temporal logic implementation of what they define as contextual privacy. Their approach uses linear temporal logic to define privacy model as agents, attributes, and messages. In addition, they model contextual integrity using roles, context, and traces. In fact one can see a similarity between the subject verb object model

(SVO) studied in [Hassan 2005] [Hassan 2006.1] [Hassan 2006.4], the work by Hruby [Hruby 2006] and the work by Barth[Barth 2008].

Worthy of mention is the work of Kowalski [Kowalski 1985], Bench-Capon [Bench-Capon 2008] and others on the limitations and possibilities of representing laws in formal logic. Our work in comparison is focused on privacy and financial laws, particularly looking at the logical, ontology, and scenario aspects. We regard the law is a large requirements set with many aspects that cannot be implemented in computer or business systems. We address aspects that can be automated.

3.5.4 Relation to URN, i*, and UML

The framework presented in this thesis can be regarded as a parallel method of requirement extraction, representation, and validation. Hence GAM can be compared to URN [ITUT 2003], i* [Yu 1997], or UML. However, we do not attempt to pursue such an endeavor in this thesis. In this section we compare, the reference model, the meta model, model, and validation mechanisms.

URN, with its GRL and UCM notations, is not only able to describe business processes but also describe goals as specified with in requirements documents. This offers the advantage of traceability from requirements to process implementations. On the other hand, our method does not offer the same capability. The formal semantics proposed in our framework, offer a great advantage since formal validations of models can be made possible. Currently

GRL's validation is limited to static validation. Some of the papers going in this direction include [Peyton 2007] [Ghanavati 2007] [Ghanavati 2009]. GAM hence offers the advantage of formal analysis and simulation of potential scenarios. GAM is also potentially able to study completeness. Such an issue may not be the main concern for GRL requirement notation.

i* , predecessor of URN, proposes an agent-oriented approach to requirements engineering centering on the intentional characteristics of the agent. Agents attribute intentional properties (such as goals, beliefs, abilities, commitments) to each other and reason about relationships. The framework is used in contexts in which there are multiple parties (or autonomous units) with strategic interests, which may be reinforcing or conflicting in relation to each other[Yu 1997]. In relation to i* we find that the GAM is suited to study legal compliance. Since an enterprise takes a centralized view of its components (agents etc), we find that the concept of agents is not needed in this case. Like GRL goals and beliefs may create traceability from requirement documents to specification, something that is not well supported with GAM. Finally, the formal capabilities of logic analysis of GAM supported by the Alloy tool are able to provide the logic advantage of reasoning. A detailed study of the advantages and disadvantages of either approach will be subject of future work.

When compared to UML, we can say that the method offers a higher layer or abstraction, yet our design artefacts are object oriented. The GAL translation

provides the semantics and provides the formalism needed for checking consistency.

3.5.5 Conclusion

Clearly, the notion of enterprise legal compliance spans several major fields of research, such as legal requirements engineering, logic and formal analysis, and enterprise modeling. The governance analysis method, which will be presented in Chapter 4, is directed at enterprise legal governance, encompassing the three mentioned areas. This chapter provided an overview of two key governance laws, namely, the privacy and financial laws. We have also presented some background on normative systems and a summary of related work. The next chapter will explain our method in detail and provide in-depth examples.

Chapter 4

Governance Analysis Method

4.0 Introduction

This thesis investigates our compliance validation framework including models, method, and tool for enterprise governance. In this chapter we will present the governance analysis method for compliance validation and the process of refining the requirements into a first order predicate logic ready for validation.

We see the need for a method and tool capable of assisting in validating the compliance between enterprise and legal requirements. We propose a method that fits into enterprise legal governance process as identified in Chapter 2. In contrast to other approaches listed in Chapter 3, our approach uses logic as a basis and proposes an extraction method using our own legal and enterprise requirements representation language.

The Governance Analysis Method (GAM) can be used to check consistency as a necessary condition for validating legal compliance. More specifically, the method guides a GO to 'partially' extract legal requirements from the law. We say 'partially' because the extraction process may not extract all the legal requirements owing to the complexity of the legal text as discussed in Chapter 2. In addition, the method includes a translation of requirements process from the law to the GAL. The GAL is read by the GAT, which produces a logic analyser model. Finally, the logic analyser model can be analysed by checking

the consistency and the scenarios. This chapter provides a detailed view of the method accompanied by small examples showing the detailed steps needed to validate compliance.

Following the spiral approach proposed earlier in this thesis, we started by defining compliance in Chapters 1 and 2. Chapter 3 complemented the discussion by presenting the background on compliance. Chapters 7 and 8 will present detailed case studies validating and testing solutions on PIPEDA and SOX examples. In proposing this method, the paradigm we follow is one of conceptualization, empirical exploration and testing.

In this chapter, section 4.1 provides an overview of our governance analysis method. This overview will discuss the activities that an enterprise GO would follow to apply the method. The remainder of the chapter will illustrate the method in more detail by presenting the sequence of steps starting with the requirements extraction presented in section 4.2. In section 4.3, we will show the generation sub-process. The validation process is shown in section 4.4. The formal semantics are shown in section 4.5. Section 4.6 will show the integrated implementation of the tool and section 4.7 will present a process for using the tool proposed in 4.6. Finally, section 4.8 will conclude with a summary of the chapter.

4.1 Overview of GAM

Our method, the governance analysis method or GAM, is composed of five major processes and uses the Governance Analysis Language (GAL) as well as the

Governance Analysis Tool GAT. The sequence is as follows: Extract and represent requirements in GAL; generate the logic model using GAL instructions as input; run the logic analyser to view results; and, apply theme filters to analyse the results.

1. *Extract and Represent requirements* in GAL is a manual activity and features an extraction of the legal requirements from the law or regulations and their representation in GAL. The process will be described in this chapter. However, Chapter 5 will provide the details on how to extract the requirements from the law using patterns. The pattern details are mapped to GAL instructions in Chapter 6.

2. Generate the logic model using GAL instructions as input is an automated step producing a logic analyser model.

3. *Run the logic analyser* to detect inconsistencies in the model. The GO can implement our proposed consistency checking techniques, namely logic, ontology, and scenario checking.

4. *Apply theme filters* to simplify results. The simplified view can show one or more classes in enterprise model and their relations

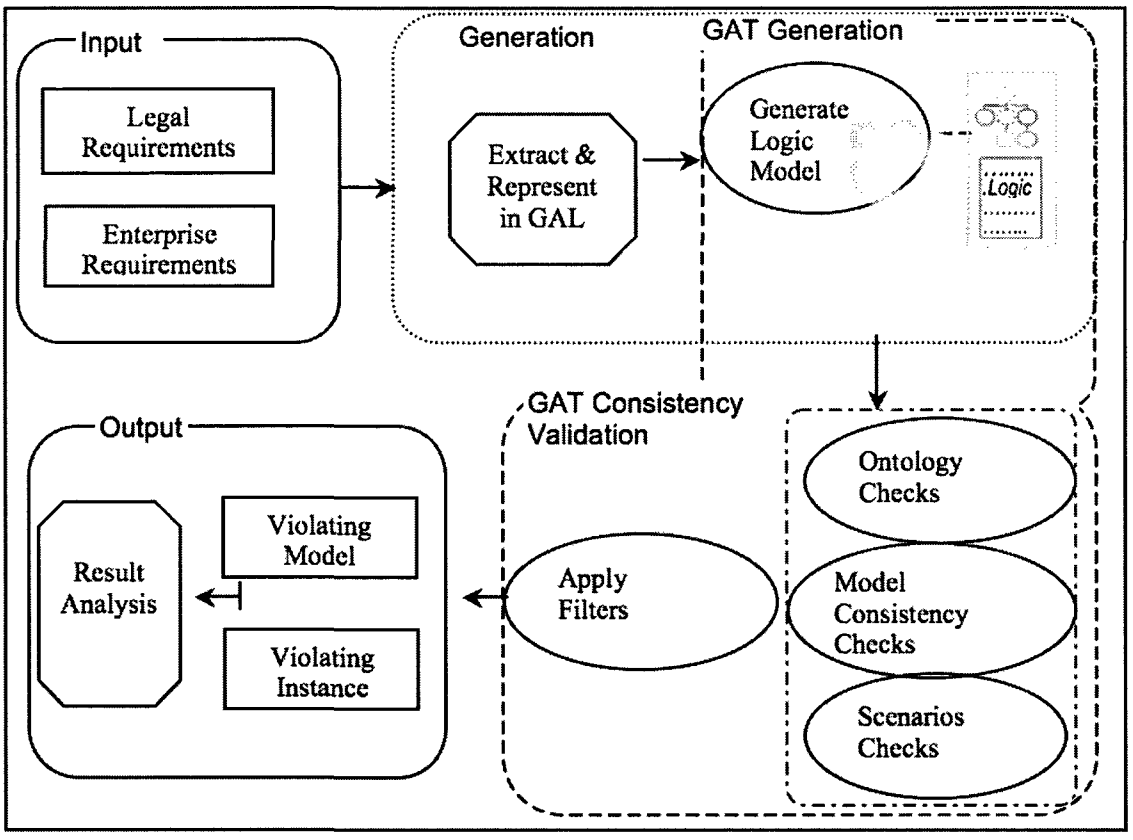


Figure 4.1: GAM Method

Figure 4.1 shows the processes of the GAM. The rounded *input* rectangle presents legal and enterprise requirements, which are the two input sources. These requirements have particular patterns that we will discuss further in Chapter 5. The *Extract and Represent to GAL* octagon illustrates the manual activity for representing high-level requirements (see Chapter 6). The ovals located within the dashed rounded boxes represent the tool engines. The tool is composed of the generation engine (GAT Generation) that reflects the requirements in ontologies and logic statements in addition to the validation

engine (GAT Consistency Validation). The consistency validation module of the tool can check model consistency, ontology, and scenario as a prerequisite for compliance.

If a counter example for a scenario exists, the tool may be able to produce a violating instance. A violating instance shows an instance of a world where the requirements are violated whereas a violating model simply indicates a logical contradiction without providing further detail. The octagons in the rounded output squares represent manual analysis by the governance officer. We will show an example of the possible analysis in Chapters 6 and 7. Table 4.1 lists the inputs and outputs of the GAM.

Table 4.1: Inputs and outputs of GAM

Type	Activity	Input	Outputs
Manual	Extract & Represent in GAL	Legal Script Enterprise Policy	GAL Requirements
Automated	Generate Logic Model	GAL Requirements	Alloy Logic Model
Automated	Logic Check	Alloy Logic Model	Violation Model
Automated	Ontology Check	Alloy Logic Model	Violation Model
Automated	Scenario Check	Alloy Logic Model	Violating Instance
Manual	Apply Filters	Result Model	Specific Model Elements
Manual	Result Analysis	Specific Model Elements	Explain Semantics of Violation Possible options for resolving Violation Tracing Cause of Violation

The first column represents the type of activity (manual, automated), the second column represents the activity name. The third column shows activity's input and the third column shows the produced output.

The output results provided by the GAM provide the means of supporting the requirements consistency validation and assisting in the provision of an assurance of compliance validation.

4.2 Generation process

This section will discuss the GAM activities a GO must follow for the purpose of validating compliance. The techniques are provided to help execute these activities. As shown in Figure 4.1, the generation process includes the manual translation to the GAL in addition to the generation of the logic model.

4.2.1 Representing requirements

Our method proposes the partial extraction and representation of the governance requirements in the form of GAL statements. The first steps in GAM's process are the manual extraction and translation. These will be further studied in Chapters 5 and 6.

Governance requirements are composed of legal and enterprise requirements and are shown on the left side of the following figure

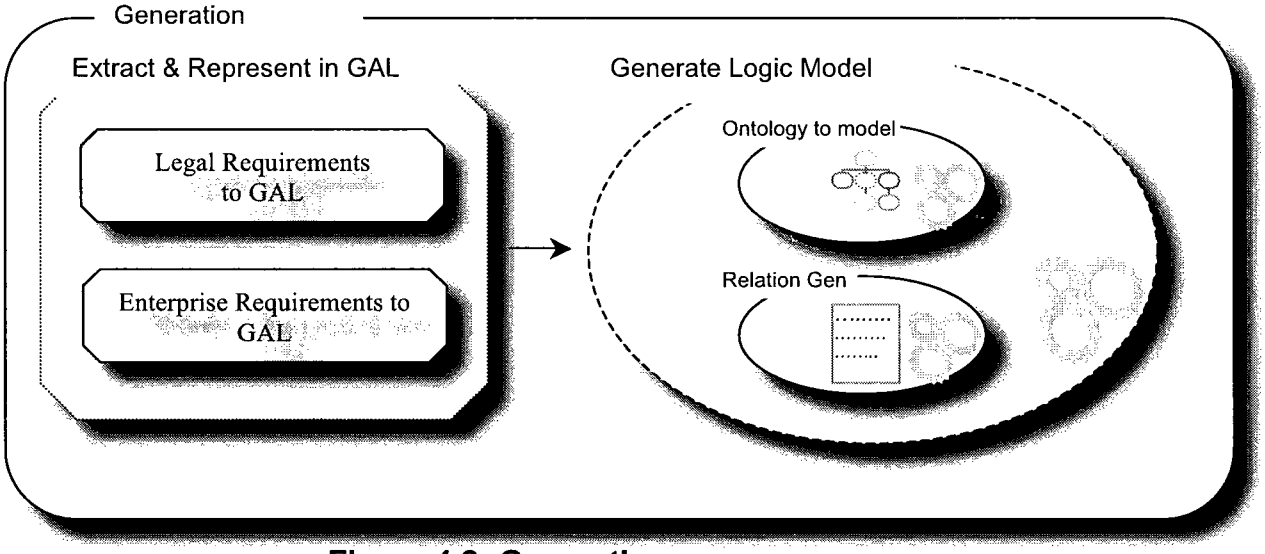


Figure 4.2: Generation process

Whenever laws change, GAL statements may have to be modified or added to represent the new types of requirements.

4.2.2 Generate logic analyser model

The *Generate Logic Model* process takes the GAL instructions as input and produces an Alloy-4 model as output. Using our user interface, a user may simply select the *Generate Logic Model* option. In addition, the tool will read the ontology requirements and specify those into logic terms. Hence both modules create a single GAL file.

Consequently, the tool will read the GAL file and generate the Alloy logic model, preparing it for consistency and compliance checks.

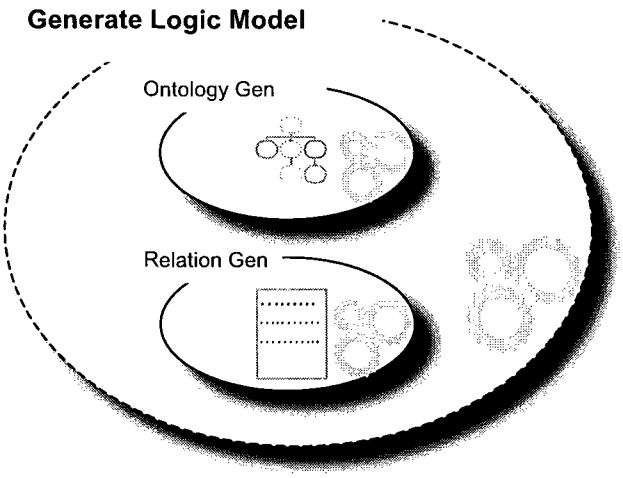


Figure 4.3: Generate Logic Model

Chapter 6 will discuss the translation process in detail. It will include a discussion of our Governance Analysis Language (GAL) in addition to providing sample logic representations.

4.3 GAT validation sub-process

As shown in Figure 4.4, the automated tool is capable of validating the generated module for consistency. After reading the input, the tool will produce a logic check that, in return, may discover the contradictions. If any contradictions are discovered, the model is considered a violating or non-compliant model. In this case, a user will need to check definitions written in GAL.

A GO can proceed to ontology and scenario checks only once the logic check has passed. After each attempt, a user may also apply filters to the results to focus the view on certain aspects of the model.

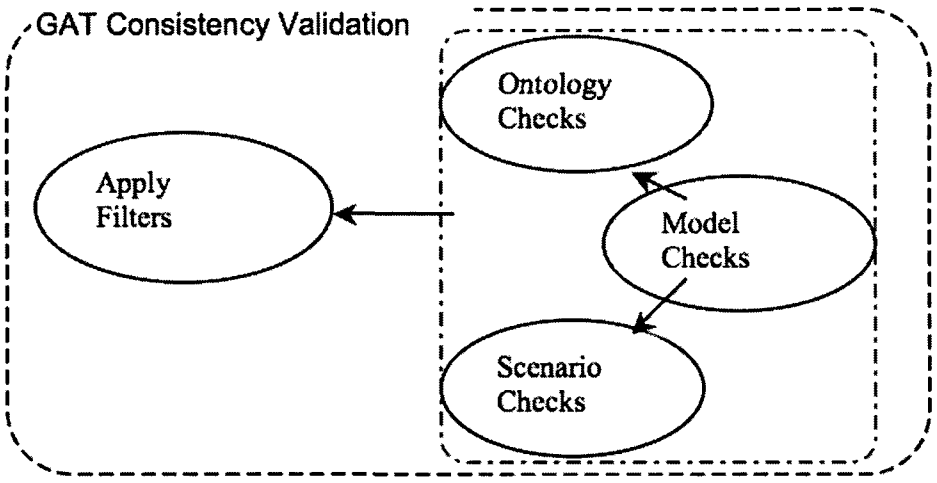


Figure 4.4: Governance analysis tool consistency validation module

Validating compliance includes scenario validation. Scenarios required by law can be checked against enterprise requirements either one at a time or in groups. Each scenario is usually composed of a group of provisions, which the GO would need to validate as a whole. The GO should also try to validate provisions based on their legal groupings which can easily be specified in GAL input files. If a provision fails to prove true, then it was found invalid in some instance. The tool will present the instance in which the violation occurred for the benefit of user analysis.

Given that the results of a model as displayed by the tool may be complex to interpret, filtering may be helpful. A filter may show certain classes or relations or a combination of both. If any of the provisions fail to validate, a GO may apply a filter to the results with the purpose of revealing the source of the problem. Not having a filter could hinder the ability of a GO to interpret the produced output.

Figure 4.5 describes in detail the process flow of the GAT consistency validation process. First the model consistency check is run, and if it fails, then there is no model that can be analysed. Otherwise, the GO can run scenario or ontology checks. For each check, the analyser can declare that the check was satisfied and declare that the formula is valid within the scope of analysis. Otherwise, if the scenario of a check fails, the analyser produces a counterexample. The GO can then apply a filter, which will limit the presentation to the classes and the relations that are considered interesting. If the tool does not indicate the exact reason for an error, the GO needs to do some further analysis.

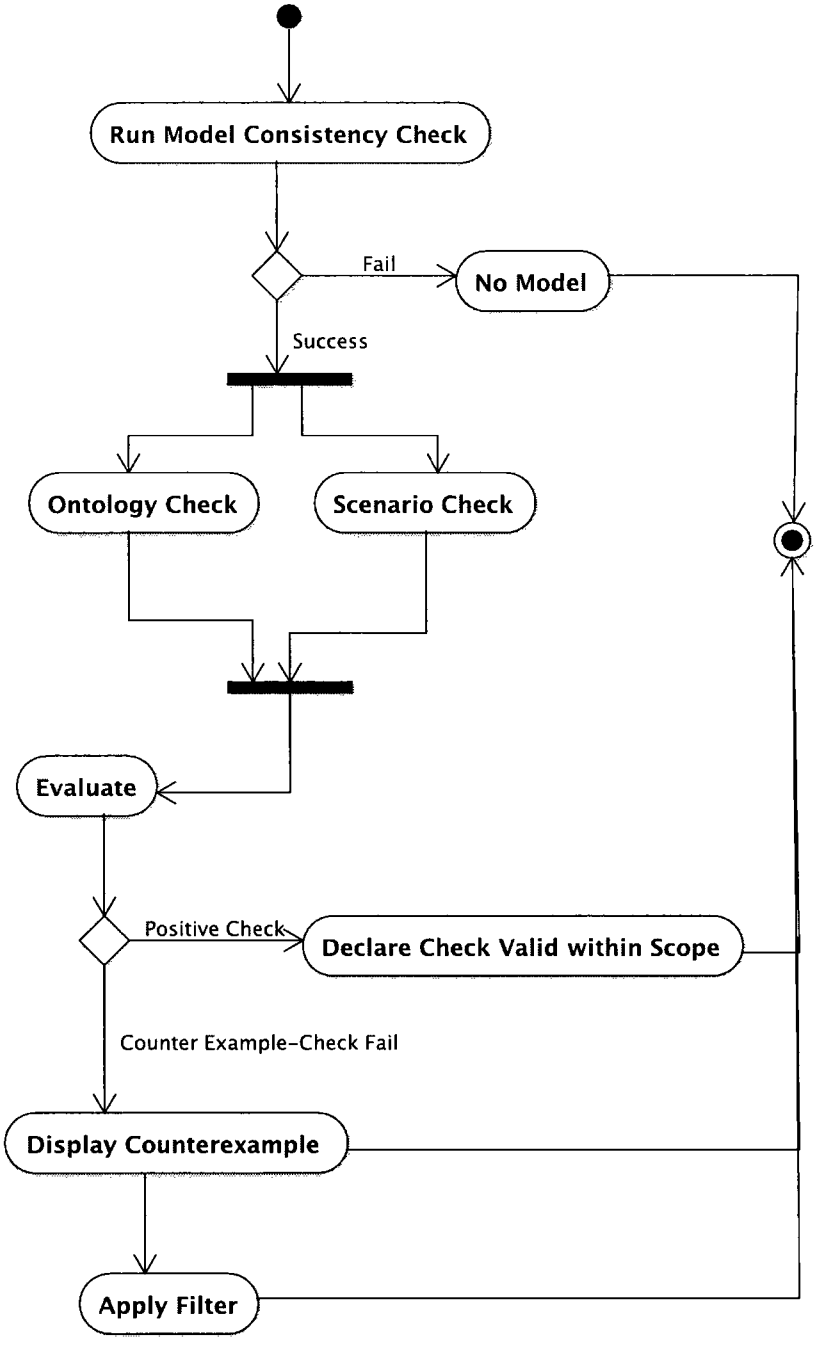


Figure 4.5: Usage flow diagram

4.4 Formal representation

In this section, we will introduce a formal representation of our formal model. In the first subsection, we will define a formal requirements system and in the second subsection, we will define consistency as the intended goal.

4.4.1 Requirements system

We base our formal definition on the semi-formal UML presentation of the combined requirements model presented in Chapter 2. We formally define an enterprise or legal requirement system to be composed of a set of formulas written in first-order logic.

The predicates in this system are the following:

1. Set P is the set of processes
2. Set A is the set of activities
3. Set R is the set of roles
4. Set U is the set of users

We will represent ontologies using sets and their relations. The relations of the formal system are also represented by means of predicates.

The predicates are:

Set X is the set of process pairs that are in separation of concerns. For example, (p,p') where $p,p' \in P$

Set D is the set of allowed delegation roles or delegated roles and their activities or processes. For example, (r,r',[a | p]) where r,r' $\in$ R and r $\neq$ r', a $\in$ A, p $\in$ P means that role r can delegate to role r' activity a or process p .

Set S is the set of activity sequence pairs. For example, (a,a') where a,a' $\in$ A. This means that activity a precedes activity a' within the enterprise process

Set RA is the set of role-to-activity assignment pairs. For example, (r,a) where r $\in$ R, a $\in$ A.

Set UP is the set of user-to-process assignment pairs. For example, (u,p) where u $\in$ U, p $\in$ P means that user u is assigned to perform process p.

Set RR is the set of role-to-role parent child pairs. For example, (r,r') where r,r' $\in$ R, r $\neq$ r' means that role r is the parent role of r'.

Set PP is the set of process-to-process parent child pairs. For example, (p,p') where p,p' $\in$ P, p $\neq$ p' means that process p is the parent process of p'.

Set RU is the set of role-to-user assignments. For example, (r,u) where r $\in$ R and u $\in$ U means that role r is assigned to user u.

Set PA is the set of process activity relations. For example, (p,a) where p $\in$ P, a $\in$ A means that a process p includes activity a.

$\therefore$ A requirements system is defined as set of formula sets.

$\Phi = \{ P, A, R, U, X, D, S, RA, UP, RR, PP, RU, PA\}$

Within the following limitations, we conjecture that a system of logic formulas Φ is able to represent requirements and offers a reasonable vehicle for validating consistency:

– Φ can be incomplete since the translation of requirements can be partial;

– Φ can contain assumptions

– Φ can include errors introduced in the translation process.

Figure 4.6 presents the flow of the GAM and shows the correlation between the formal representation in section 4.5 and the rest of the process. The diagram shows that requirements come from two sources: the law and the enterprise; they are extracted and presented in GAL which is transformed into the logic model Φ which represents the ontology in addition to other relations. Finally, the logic check tests the model Φ, and the results are displayed after possible filtering.

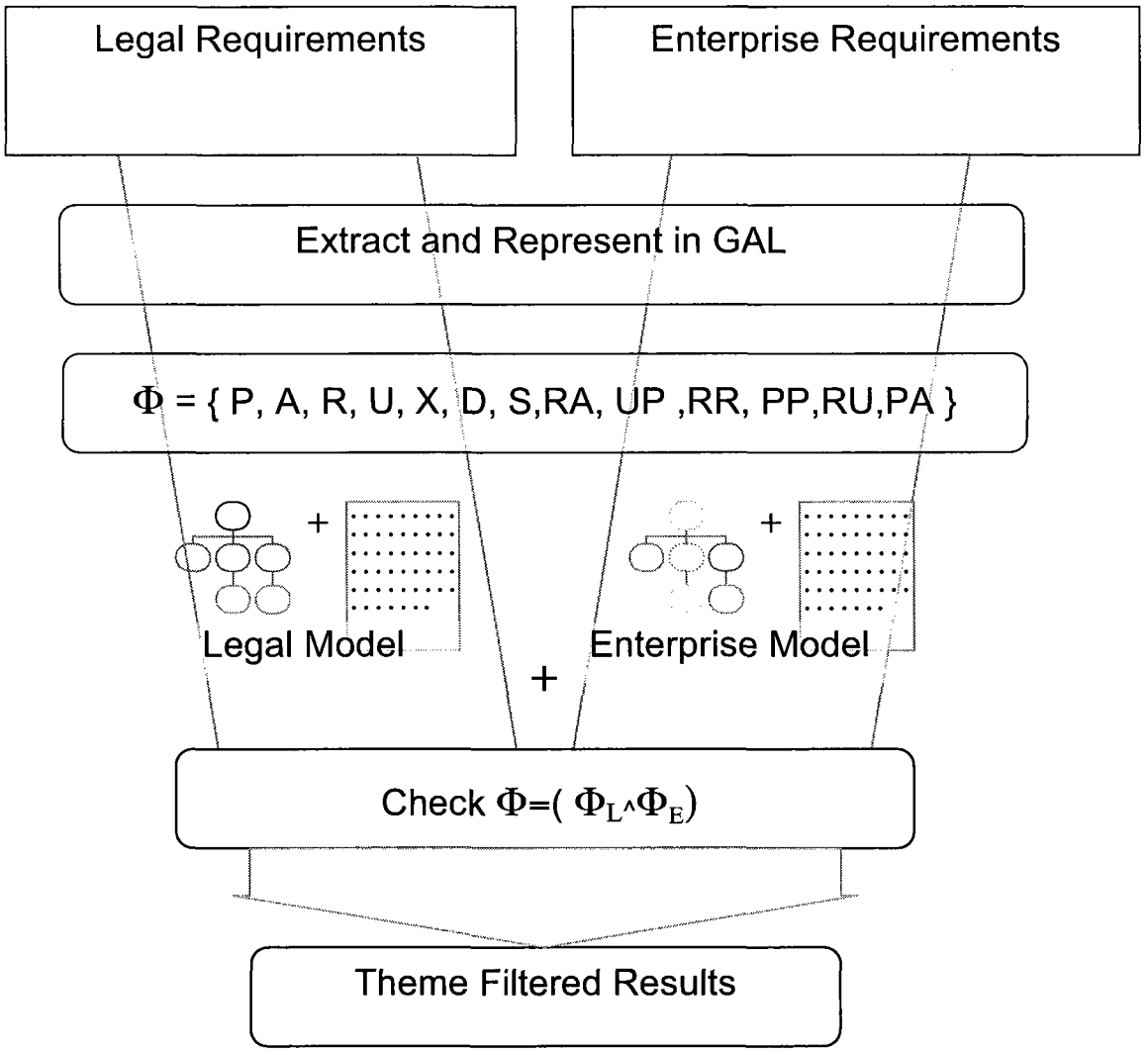


Figure 4.6: Formal representations

4.4.2 Consistency

Formally, the legal requirements will be represented by a set of formulas ΦL whereas the Enterprise requirements will be represented by a set of formulas ΦE.

In this thesis we provide three techniques to assist in checking consistency:

1. Model consistency check: A system written in first order logic consisting of a set of formulas Φ is consistent (written Con(Φ)) if and only if it has no contradictions; otherwise Φ is inconsistent, which is written Inc(Φ). To validate Con(Φ) we will use a logic analyser that depends on SAT solvers.

   Further, two systems are considered consistent if the conjunction of their formulas $(\Phi_1 \wedge \Phi_2)$ is consistent. Hence, an enterprise system ΦE is consistent with a legal requirement system ΦL if (Con(ΦE ∧ΦL)). A system (ΦE ∧ΦL) can be validated by a logic analyser to have no contradictions.

2. Ontology check: An ontology check insures that the enterprise ontology has the elements and relations as defined by requirements in the law. There are two types of requirements: requirements of structure and requirements of process. The check ensures that the enterprise respects organisational and process structures, including activity sequences. Laws do not normally check for instance names, however enterprise requirements may do so.

Structure: $$\left\{ \begin{array}{l} \left\{ \begin{array}{l} \forall r \in R \\ R \subset \Phi L \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} r \in R' \\ R' \subset \Phi E \end{array} \right\} \\ \left\{ \begin{array}{l} \forall rr \in RR \\ RR \subset \Phi L \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} rr \in RR' \\ RR' \subset \Phi E \end{array} \right\} \end{array} \right\}$$

$$\text{Process: } \left\{ \begin{array}{l} \left\{ \begin{array}{c} \forall p \in P, a \in A \\ P, A \subset \Phi L \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} p \in P', a \in A' \\ P', A' \subset \Phi E \end{array} \right\} \\ \left\{ \begin{array}{c} \forall pa \in PA \\ \forall pp \in PP, s \in S \\ PA, PP, S \subset \Phi L \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} pa \in PA' \\ pp \in PP', s \in S' \\ PA', PP', S' \subset \Phi E \end{array} \right\} \end{array} \right\}$$

$$\text{Instances: } \left\{ \begin{array}{l} \left\{ \begin{array}{c} \forall r \in R, \forall u \in U, \forall a \in A \\ R, U, A \subset \Phi L \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} r \in R', u \in U', a \in A' \\ A', R', U' \subset \Phi E \end{array} \right\} \\ \left\{ \begin{array}{c} \forall up \in UP, \forall ru \in RU \\ UP, RU \subset \Phi L \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} up \in UP', ru \in RU' \\ UP', RU' \subset \Phi E \end{array} \right\} \end{array} \right\}$$

3. Scenario Checking: Positive and negative scenarios can include ontology checks and checks of relations, such as separation of concerns and delegation rules. A positive check seeks to assert that a certain scenario is possible, whereas a negative check tries to ensure that a certain scenario is not possible. It is also possible to have a combination of positive and negative checks, which seek to assert the positive part of the formula and ensure that the negative property does not occur. In formulas 1 and 2 below, both checks are represented in logic formalism:

$$\text{Positive: } \left\{ \begin{array}{c} \forall S, RA, X, D \subset \Phi L \\ \forall S', RA', X', D' \subset \Phi E \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} S \subset S', RA \subset RA', \\ X \subset X', D \subset D' \end{array} \right\}$$

$$\text{Negative: } \left\{ \begin{array}{c} \forall S', RA', X', D' \subset \Phi E \\ \forall S, RA, X, D \not\subset \Phi L \end{array} \right\} \Rightarrow \left\{ \begin{array}{c} S' \cap S, RA' \cap RA \equiv empty \\ X' \cap X, D' \cap D \equiv empty \end{array} \right\}$$

Combined scenarios to be validated are represented by joining positive and negative checks. For example assume that a GO wants to assert two requirements. One checks if role-activity relations RA_1 is matched in the

enterprise. The second checks that the enterprise requirements do not include a certain set of role-activity relations RA_2. This requirement is presented in logic as follows: $\begin{Bmatrix} \forall RA' \subset \Phi E \\ \forall RA_1 \subset \Phi L \\ \forall RA_2 \not\subset \Phi L \end{Bmatrix} \Rightarrow \begin{Bmatrix} RA_1 \subset RA' \\ RA' \cap RA_2 \equiv empty \end{Bmatrix}$

4.4.3 Formal examples

For the benefit of providing some examples of requirements systems we present some formal representations of requirements

1. The signingOfFinancialCompliance requires board approval

   Logic: (BoardApproval, SigningOfFinancialCompliance) $\in$ S

2. A user can act or not act in a particular role(s), e.g. JaneSmith can not act as a manager of finance

   Logic (JaneSmith, FinanceManager) $\notin$ RU

3. Check if a privacy process has an audit sub-process.

   (Privacy,PrivacyAudit) $\in$ PP

4. A process includes the specified activities

   Logic: (Privacy,PurposeListCollection) $\in$ PA

5. Check if the finance department has an audit division

   (Finance,Audit) $\in$ RR

6. Separate Financial Audits from Privacy Audits (FinancialAudit, PrivacyAudit) $\in$ X

4.5 Integrated implementation environment

In this section we describe the GAT modules, we also describe change management process.

4.5.1 GAT modules

Figure 4.7 shows our tool and its elements in schematic form. We have extended the tool's implementation based on an earlier version presented in previous work [Hassan 2005].

It shows data flows in addition to module composition of the GAT. On the left side of the figure, the main three aspects of a governance office may be observed, namely legal, privacy, and financial offices. The legal, enterprise and financial requirements coming from the governance office are represented in the three squares.

The logic generator and the logic analyser are the two main modules of our tool. The logic generator translates legal, enterprise, and other governance requirements into a logic model. In addition, the logic generator uses theme files that will assist in the analysis. The logic analyser consumes the logic model and theme files, and is capable of providing results in forms of example violations, counter examples, logic assertions, and aspect analysis. The logic analyser can also help focus the analysis on certain aspects as defined by the theme file.

The governance officer combines requirements from the Legal, Financial, and Privacy laws.

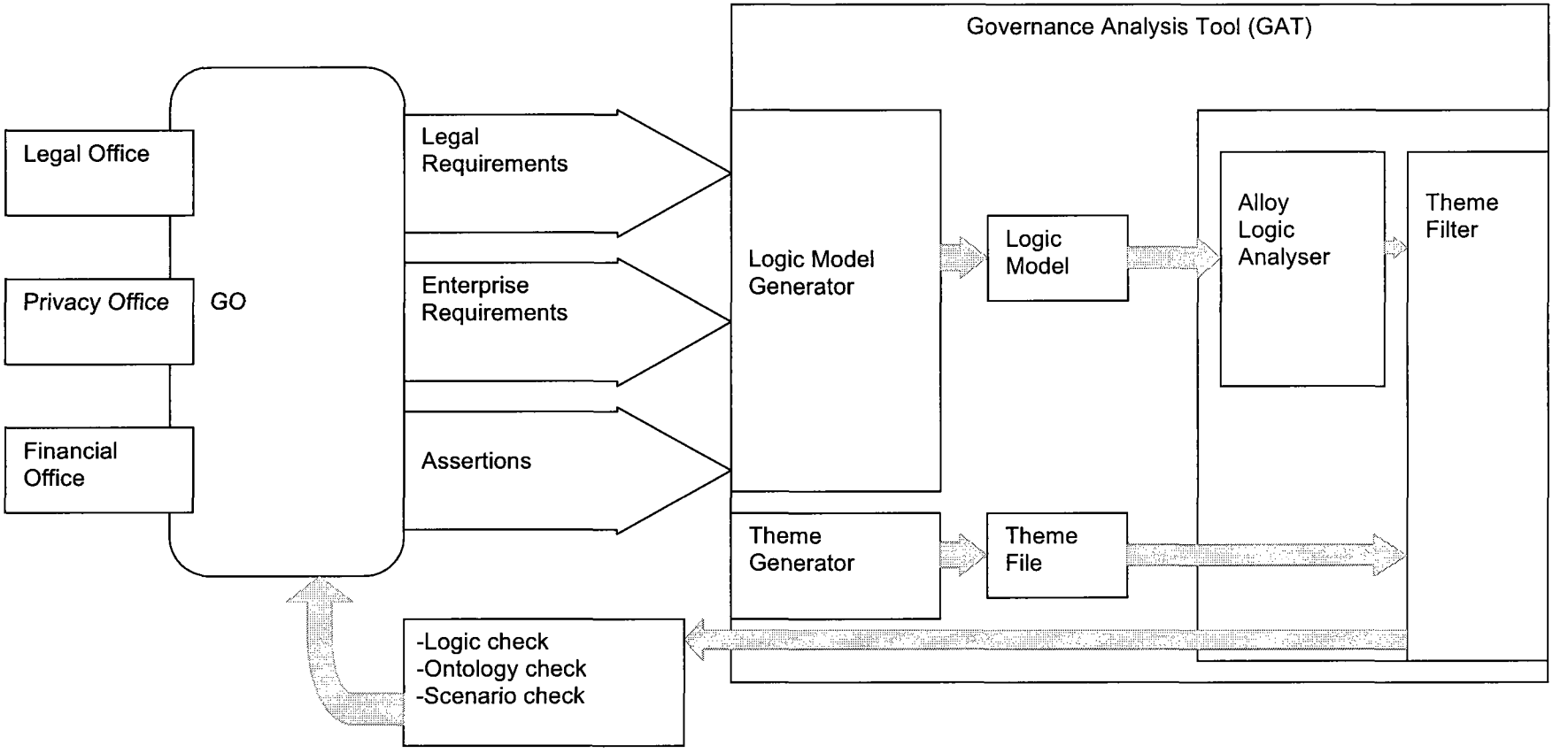


Figure 4.7: Inputs and Modules of GAT

4.5.2 Change management process

Changes in requirements must involve partial or total re-validation. Figure 4.7 shows the flow of activities in this case. From the diagram, one can see that there are two types of requirement changes: from the enterprise, or from the law.

Each of these changes can be either incremental, or be a change in the relational structure. An incremental change simply implies that new enterprise policies or provisions in the law have been added. If so then the change is relatively simple. The new statements need to be added to the GAL model. The model then needs to go through the compliance validation process as per the method. However, if the change is structural, in other words, if there are changes in relations in the model or if classes were added to the model, then this means that the tool must be changed and the patterns may need to be revisited. Furthermore, the whole validation process needs to be repeated.

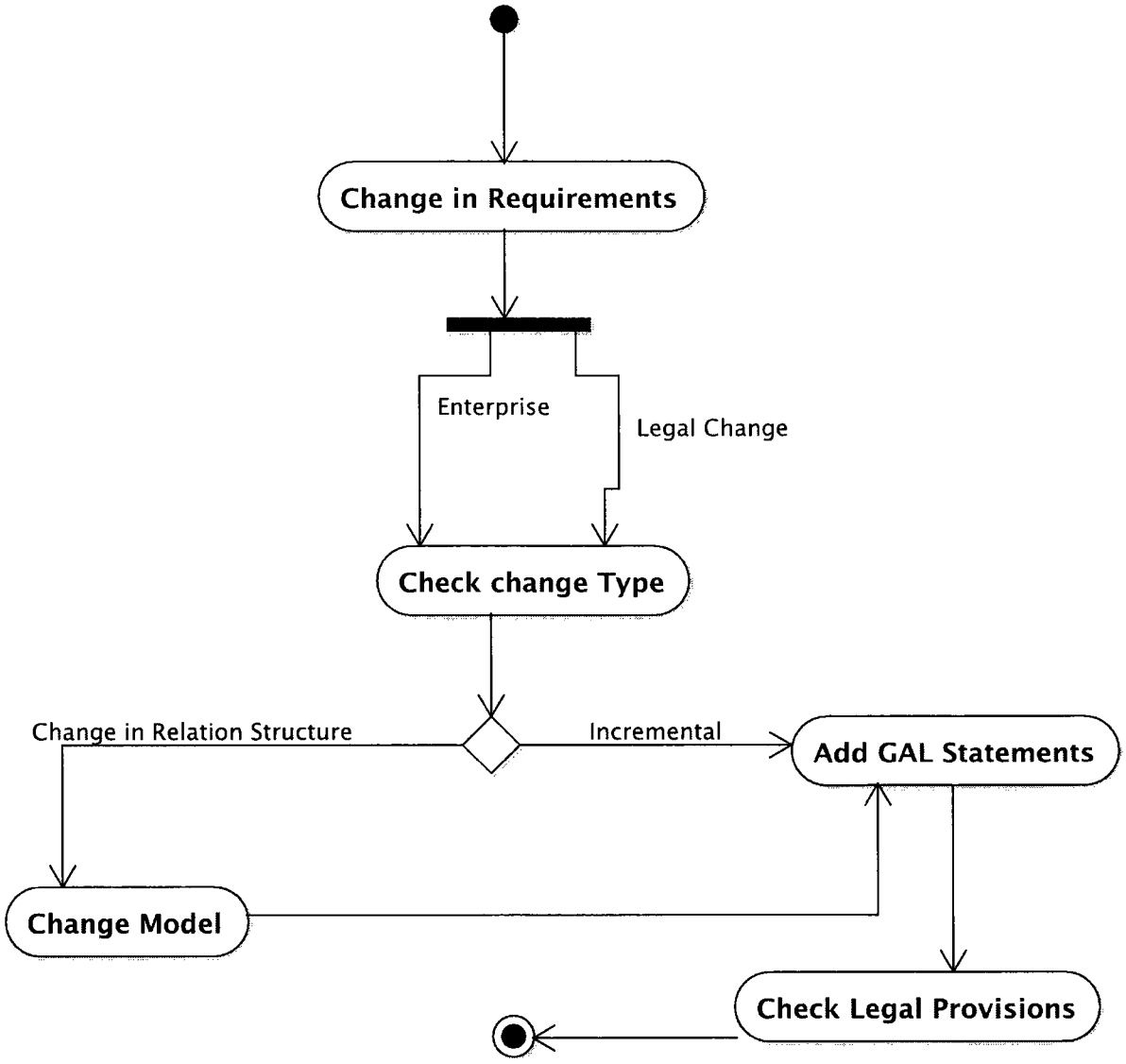


Figure 4.8: Requirements change Process

4.6 Summary

We have developed a compliance framework, including a high-level method with an accompanying language and tool.

In summary, a governance enterprise system is considered formally compliant if its requirements are consistent and complete with respect to legal requirements.

Our semi-automated process for verifying compliance follows these major steps

1. Extraction and translation of legal and enterprise requirements
2. Generation of a logic model
3. Logic analysis.

This is a repeatable compliance process. The process starts when enterprise law is extracted and translated into our logic-based legal requirements specification language. The enterprise specification is also translated into the same governance language,

The tool we have implemented uses our Governance Analysis Language, shown in chapter 6, as input and generates a logic model representing the requirements combined with enterprise specifications. The tool then passes the constructed model to the Alloy logic analyser. During analysis, the tool visualizes complex enterprise entities including processes, departments and roles, user assignments and their relations. Theme filters can project this complex state of affairs into various visual displays for detailed analysis. The

analyser output displays a complete violating model illustrating instance objects and associated relations. Output can be stored in various formats. In the following two chapters, we will demonstrate the extraction and translation of the law into GAL language.

Chapter 5

Extracting Legal Requirements

5.0 Introduction

As mentioned in Chapter 3, extracting requirements from legal text is a challenging task. Written in natural language, such text is not practical for computer interpretation [Brodie 2006] [McCarty 2007]. Legal text in governance laws is no exception. Privacy and financial laws are complex and include a large number of internal and external references, dictionary and ontology definitions, obligations, permissions, process definitions, conditional statements, and others. The above-mentioned complexity indicates the importance of the extraction method in any framework that claims to address compliance, and ours is no exception. Hence, this chapter describes our extraction method.

Our approach differs from the existing methods, presented in Chapter 2. We argue that governance laws are often procedural and specify conditions for transactions in addition to goal settings. Contrary to our view, existing work in this area approaches the law as a set of obligations and modalities or access-rights. In contrast, we argue that the governance laws are often procedural and specify conditions for transactions in addition to goal settings. We propose the use of first-order logic with embedded ontology definitions to define the legal and enterprise requirements. Our proposed translation process includes the extraction of the enterprise ontology (structure and process) along with access-rights, the delegation of authority and the separation of concerns provisions.

To support the process of translating laws into the GAL, we propose the use of extraction patterns based on an extraction model. The model is used to define syntax and semantics. As previously introduced in Chapter 1 and presented in Chapter 2, the extraction model provides the structure and relations of the set of patterns. GOs will use the patterns as guidance as during requirement extractions.

This chapter will be divided in five sections, where section 5.1 will present the existing requirements extraction methods. Section 5.2 will present the extraction pattern definitions and profiles, while section 5.3 will illustrate our proposed extraction method. A short discussion of our method will follow in section 5.4 and an application example of the method will conclude this chapter in section 5.5.

5.1 Existing methods

Several researchers have proposed methods for the extraction of legal requirements. In particular, authors of [Breaux 2006] are working on a method for the extraction of rules and obligations from regulations and legal text using automated tools. Their work focuses on building taxonomy of legal expressions and extracting obligations and rights into access control statements. Their access-rights are defined as permissions to perform actions, which are similar to deontic permissions. In contrast, our work distinguishes between requirements suggesting access-rights from those specifying enterprise processes and related activities. Contrary to the GAM, their method does not consider the business

processes in their requirements extraction processes. Their statement syntax is reflective of access-control requirements rather than regulatory compliance requirements. Their patterns represent activities where a subject performs an action on an object.

Darimont et al. [Darimont 1997] apply a methodology to model regulations. Their principal method depends on the transformation of legal documents into sets of goals, objects and threat models. Their approach is based on the analogy between regulation and requirements documents. Instead, our approach is based on a process concept; the goals are translated into elementary statements, and threat models into scenarios to be checked.

The OMG Regulatory Compliance Alliance (recently renamed the GRC Round Table) worked on standard representations of the regulatory documents. Their method aims at providing a dynamic mapping between the regulations and the organizational policies. Our method does not create a mapping: rather, it suggests combining the regulations with the organizational policies.

An ontology-based requirements extraction approach is offered by [Lee 2005] to capture and model the correlations between the attributes of certification and the accreditation requirements in the regulatory documents. Furthermore, work on a layered ontology of law was also presented by [Breuker 2004]. Their premise is the belief that law is driven by common world concepts and words, thus they include in ontology concepts, such as agent, action, organisation, in addition to legal concepts, and so on. [Ghanavati 2007] propose a compliance framework

where legislation, policies, and business processes are modeled with the User Requirements Notation (URN) [ITUT 2003]. Within their framework, traceability between the various models and source documents is provided. Their work adopts the business process as a main artifact for access right control and provides a formal representation through Use Case Maps. There are two distinguishing features for our method, the first is the use of logic representation of requirements; the second is the inclusion of processes rather than access control or deontic concepts.

5.2 Extraction patterns

Requirement Extraction (RE) is concerned with extracting the relevant compliance data from the law and the enterprise. For instance, a typical RE task might be to find management rules reported in the law .

A key component of any RE system is its set of extraction patterns (or extraction rules) that is used to extract from each requirements-source the compliance information relevant to the enterprise. Because writing useful extraction patterns is a difficult and time-consuming task, several research efforts have focused on learning the extraction rules from training examples [Jackson 2003] [Biagioli 2005]. We follow a similar method by defining extraction patterns based on the examples we have studied and according to the extraction model we have defined.

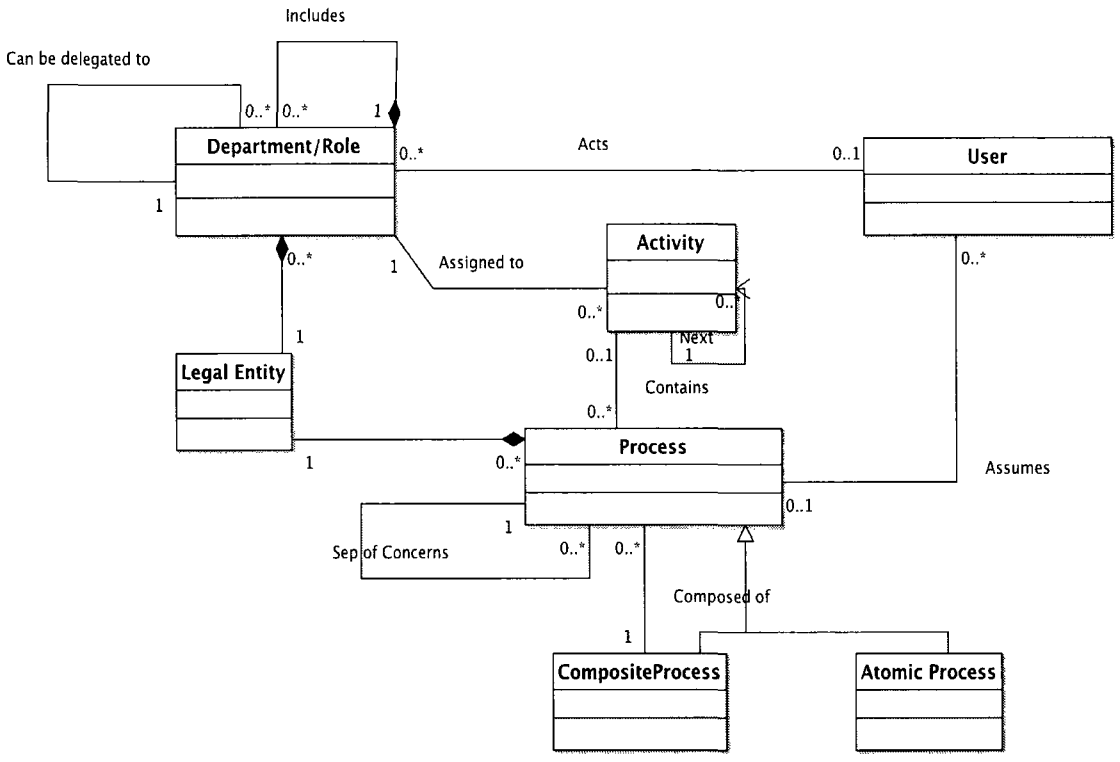


Figure 5.1: Base Model for extraction patterns

In this section, we provide the extraction patterns we identified in our analysis of the governance laws for the purpose of translation into a formalised language. Note that our discussion of extraction patterns is not aimed at automating the extraction process. Rather, it is aimed at providing guidelines to a GO intending to translate law and regulation requirements into a formalised language.

We start this discussion by showing the adopted requirements extraction model previously presented in Chapter 2 as the combined model. This extraction model is used to represent patterns relations and semantics.

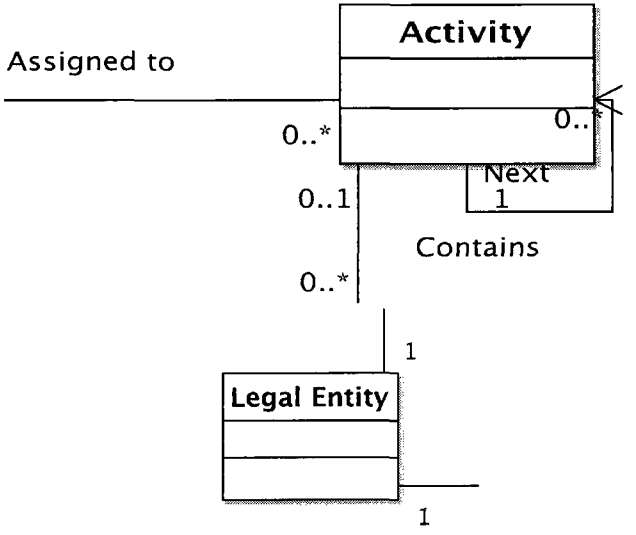
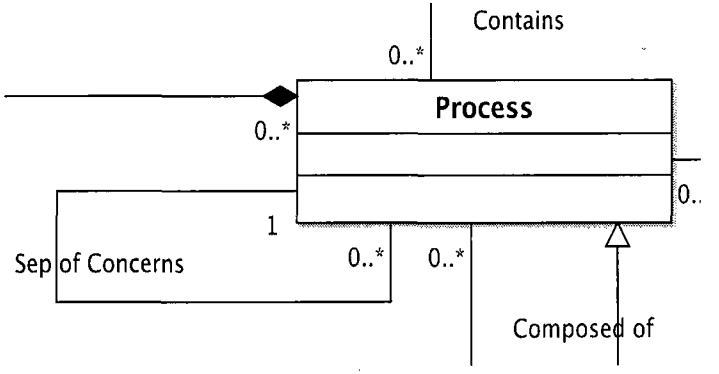
5.2.1 Pattern list

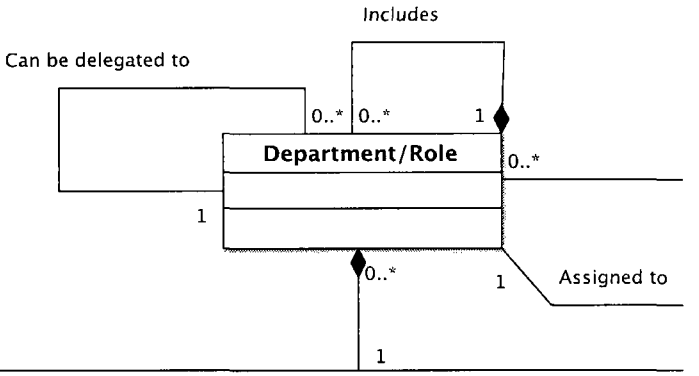
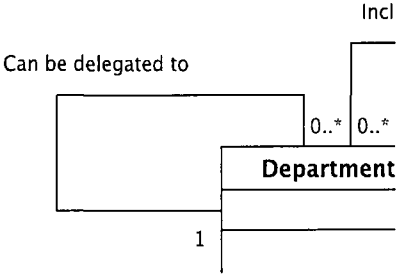
The patterns illustrated by Figure 5.1, a replica of Figure 2.12, are categorized by class, relation, and high-level classifications. The "class" category groups requirements that can be represented using classes. The "relations" category includes patterns that are represented using relations. The high-level grouping represents patterns including higher-level requirements, usually declarative in style or composed of other requirements.

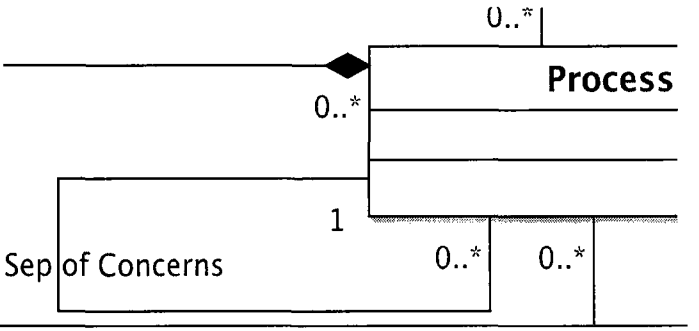
The patterns that we have studied so far are the following: Activity, Legal Entity, Process, Role, Assignment, Delegation, Separation of Concerns, Composite, Declarative, and Definition type patterns. The class category includes (Activity, Legal Entity Process, and Role).

The relations category includes (Assignment, Delegation, and Separation of Concerns). Finally, the higher-level category includes (composite, declarative, and definition patterns).

Table 5.1: Pattern definitions

Classes	**Activity**	A set of one or more activities that are a part of the process and can be assigned to roles. An activity Pattern may also suggest a particular sequencing . A conditional activity is a specific type of activity that includes a check followed by possible options. Relations: -Contains -Next	
	Legal entity	A legally recognised entity. Such a pattern is usually helpful if the requirements include multiple parties.	
	Process	Pattern suggesting a process, which may contain other processes or activities. A process pattern can define a process-to-activity relation. Relations: -Contains -Composed Of	

Role	A hierarchy of roles and possible assignment to activities. . Relations: -Includes	
Assignment	A subject, such as a role or a group of users can receive an assignment to access an activity or a process. The assignment can be explicit by stating that a subject is assigned to an activity or implicit by stating that the subject has the right to access an activity. .	
Relation	Relations: -AssignedTo -Assumes -Acts	
Delegation	A pattern representing a right of delegating or a delegation of an activity from one role to another. Relations: -Can be delegated to	

	Separation of concerns	Processes in separation of concerns cannot be accessed concurrently by the same role or users. Relations: -Separation of concerns	
	Composite	High-level requirements that can be represented using several elementary patterns such as processes, roles, or activities.	
Higher Level	**Declarative**	Usually a desired state of affairs rather than a process. Usually stated as "An entity is responsible for implementing or following a specific process".	
	Definition	This pattern proposes equivalence relations where a class is made equivalent to another. A definition may simply provide a fact that can be implemented through one or more or relations.	

5.2.2 Pattern profiles

In this subsection, we will present a profile for each pattern. Each profile will include the pattern name, enterprise or legal model elements (classes), and an example along with descriptions.

The classes are taken from the extraction model illustrated in Figure 5.1.

Activity

Pattern Name: Activity	
Classes	Activity, role
Structure	An action-verb or a condition
Relations	Next, assigned to, contains
Example	*The chief financial officer signs stock certificates* Role: Chief financial officer Activity: Signs stock certificates
Details	An activity is an action verb or a condition that can be sequenced using the Next relation and can be assigned to a role. An activity is contained in a process. .

Legal Entity

Pattern Name: Legal Entity	
Classes	Legal entity

Structure	A name of a legal body
Relations	None
Example	*ISSUER—The issuer as defined in section 3 of the Securities Exchange Act of 1934 (15 U.S.C.78c): the securities are registered under section 12 of that Act.*
	Name: The above is a definition of a given company.
Details	The example above shows the legal definition of a publicly owned company and refers to it as the issuer, because it issues stock certificates

Process

Pattern Name: Process	
Classes	Process, activity, composite process, atomic process
Structure	A name of a business function that is composed of: - Other business processes or - Activities
Relations	Assumes, separation of concerns, composed of, contains
Example	*The audit process includes:* *a) Annual financial statement audit* *b) Statutory audit in the US* *The audit committee is responsible for signing the audit process.* Name: Audit Process Composedof: Procedural Reviews, Testing,

	Information Systems Processes. Activities: Audit annual financial statement, execute statutory US audit.
Details	A certain user assumes a business process containing activities and composed of other processes or activities. A process may relate to other processes by separation of concerns.

Department/Role

Pattern Name: Department/Role	
Classes	Department/Role
Structure	A name of an organizational unit or a role with optional included roles or divisions. The pattern may be presented in conjunction with other patterns such as assigned activities
Relations	Contains, can be delegated to, AssignedTo
Example	*Financial admin assistants receive loan applications. Privacy Division includes a privacy audit unit.* Names of roles/Departments: Financial Admin Assistants, Privacy Division Includes: Privacy Audit is a subdivision of Privacy Division AssignedTo: receive loan application activity is assigned to role Financial Admin Assistants.
Details	A department can contain other departments. A role can have sub- roles and be assigned to activities. Certain named individuals can act in a role. A role can be delegated or prevented from being delegated to another role. .

Assignment

Pattern Name: Assignment	
Classes	User, Activity, Role
Structure	A subject is assigned to an activity or a process or A subject has the right to be assigned to an activity or process
Relations	AssignedTo, Assumes
Example	*An administrative assistant has the right to review financial information.* Subject: administrative assistant or role Activity: review financial information
Details	An assignment can provide a user or role with the right to be assigned to an explicit assignment to an activity

Delegation

Pattern Name: Delegation Right	
Classes	Role, Activity
Structure	A subject has a modality to delegate role's assignedTo activities to another role
Relations	AssignedTo
Example-A	*A CPO is allowed to delegate compliance approval.*

	Role: CPO Activity: delegate compliance approval
Example-B	Another example of the same pattern may suggest that: *A Financial Officer delegates his ability to sign privacy compliance statements to a Chief Executive Officer* Role: Financial Officer, Chief Executive Officer Activity: Sign Privacy Compliance
Details	A role allowed/denied the right to delegate his/her assignedTo activities.

Separation of Concerns

Pattern Name: Separation of Concerns	
Classes	Process
Structure	A process is separated from another process.
Relations	Contains, Assumes
Example	*No activity sharing between marketing and finance* Process-A: Marketing Process-B: Finance
Details	That a process must be separated from another means that all contained processes of the first process are not to share activities with the second process. Furthermore, any user that assumes the first process is not to assume the second process. .

The next three patterns are at a higher level and are thus not represented using specific classes in the requirements model illustrated in Figure 5.1. An extensive study of higher-level requirements is left to future work.

Composite

Pattern Name: Composite	
Classes	All
Structure	A requirement name is satisfied if the following processes or structures are followed
Relations	All
Example	*Privacy accountability is achieved by assigning a privacy officer with a privacy process that includes an audit* Requirement : Privacy Accountability Translation: (1) Privacy Officer role, (2) person acting in the role, (3) an audit process is a sub-process of a larger (4) privacy process.
Details	A requirement is achieved through a set of other lower level requirements. Composite requirements usually specify a mixture of scenarios and possibly referring to entities that need to be defined. A composite pattern can be rewritten in several statements.

Declarative

Pattern Name: Declarative	
Classes	All
Structure	A requirement name is satisfied

Relations	All
Example	*Limited Control: All Activities are controlled.* Requirement Name: Limited Control Process: validate the all activities have a modality of access by subjects
Details	The high-level requirement is defined as a goal to be achieved. The goal or state can be achieved through a process or a particular structure.

Definition

Pattern Name: Definition	
Classes	Role, User, Process, Activity
Structure	A class instance is equivalent to another instance
Relations	None
Example	*An exemption can be sickness, long-term disability. or if subject is a minor* Activity: exemption, sickness, long-term disability, subject is minor
Details	Definitions often suggest connections between roles, activities or conditions. They may equate a role with a process or another role and may provide a composition or inclusion relation. Their importance lies in their capacity to often detail ontologies.

5.3 Extraction method

We will now define a pattern-based extraction method to help us extract legal requirements from the law. The method is straight-forward and consists of reading the legal text and looking for the class and relational patterns and

extracting their elements. However, the composite and declarative patterns need to be decomposed.

Proposed method:

1. Classification: The legal text needs to be classified with the above listed patterns. For example, a governance officer may need to extract activities, assignments, delegations, definitions, enterprise entities, processes, roles and separations of concern. Some patterns define assignment or sequencing relations such has belongs-to, delegate-to, etc. These relations are defined in the model explained in the previous section: Roles-Activities, Processes-Activities, Activities-Activities, Processes-Processes, Enterprise Entity-Processes, and Roles.

2. Composed or declarative higher-level patterns can contain multiple rules or lower-level provisions. This decomposition will be left to the experience and the knowledge of the GO. One can take the accountability pattern for example, which can be decomposed into several other simple patterns.

5.4 Discussion

In contrast to other grounded theory methods suggested by [Breaux 2008] and others, we correlated an enterprise model to a legal requirements model, as previously seen in Chapter 2, section 2.5. We found that governance laws can be captured using our extraction model. This method combined the governance and the enterprise legal model and defines the meta-relations of several patterns based on ontology and legal requirements. We then provided a method to assemble the legal requirements and governing enterprise artifacts.

We believe that any method of capturing laws needs to be able to trace legal requirements into enterprise artifacts. Should this not be the case, referential integrity will be compromised. In contrast to related research, we believe that our approach incorporates the enterprise governance models and implementation artifacts.

Related research that views the law as deontic statements solely represents requirements in terms of obligations and rights [Dubois 1994]. In contrast to this type of approach, we translate obligations into process definitions and access-right relations.

5.5 Examples

In this section, we will provide several examples taken from SOX, Instruments 52–111, and PIPEDA. Each example will present a text description. We then discuss the patterns that we extracted from the text. Please note that the examples are taken from the given laws; however, they may have been slightly modified or abbreviated for the purpose of facilitating the discussion.

5.5.1 SOX-Section-2

Audit (3) AUDIT COMMITTEE. *The term "audit committee" means a committee (or equivalent body) established by and amongst the board of directors of an issuer for the purpose of overseeing the accounting and financial reporting processes of the issuer and audits of the financial statements of the issuer, and if*

no such committee exists with respect to an issuer, the entire board of directors of the issuer.

In this requirement, there are several role patterns suggested. The role of the audit committee is being defined, it also mentions that it is a sub-group of the board of directors, which is another role. Finally, it shows that the board of directors belongs to the main role department/role called issuer.

There is another pattern in the SOX example, which is the activity pattern labeled "audits financial statements". This pattern suggests that the role of the audit committee is assigned to the task of signing financial statements.

Note that there may be other provisions affecting these patterns and hence further refinement or and extension of the patterns may be described in other provisions.

5.5.2 Instruments 52–111

Canada's financial compliance law "Instruments 52–111" defines internal control audit report processes and structures using the following requirements:

52–111 – *Internal control over financial reporting is a process designed by, or under the supervision of, the issuer's chief executive officer and the chief financial officer, or persons performing similar functions, and is effected by the issuer's board of directors, management and other personnel, in order to provide reasonable assurance regarding the reliability of the financial reporting and the preparation of financial statements for external purposes in accordance with the*

issuer's generally accepted accounting principles and includes those policies and procedures that:

(a) pertain to the maintenance of records that in reasonable detail accurately and fairly reflect the transactions and dispositions of the assets of the issuer,

(b) provide reasonable assurance that transactions are recorded as necessary to permit preparation of financial statements in accordance with the issuer's GAAP, and that receipts and expenditures of the issuer are being made only in accordance with authorizations of management and directors of the issuer, and

(c) provide reasonable assurance regarding the prevention or timely detection of unauthorized acquisition, use or disposition of the issuer's assets that could have a material effect on the annual financial statements or interim financial statements;

Under the instruments example, there are two kinds of patterns (process and activity) with several instances. We start with the following:

Process Pattern: a process is labelled Internal Control over Financial Reporting (ICFR).

Assignment Pattern: The ICFR process is assumed by the CEO and CFO.

Activity Pattern: The ICFR process includes two activities, namely overseeing the accounting and financial reporting processes. Both activities are assigned to

Roles CEO and CFO. In addition, the ICFR includes an activity labelled "Record Maintenance". Validating transactions and signatures are two other activities that are included in the ICFR. Validating an authorised acquisition, use, and disposition are three other activities that are also included in the ICFR.

5.5.3 SOX section 404

A rule under SOX suggests denying shared access to certain activities or processes. The two examples are: the receipt of goods is segregated from purchasing and supplier master data, and the requisitioning is segregated from purchasing.

These two requirements fit the separation of concerns patterns because they clearly illustrate that the processes mentioned cannot be accessed by the same user.

5.5.4 SOX section 202

As provided in SOX and SEC's rules, the Committee may delegate either type of pre-approval authority to one or more of its members. The member to whom such authority is delegated must report, for informational purposes only, any pre-approval decisions to the Committee at its next scheduled meeting.

In this example, that the law is specifying a certain role's ability to delegate a single activity. Here, we have the audit-committee that can delegate its pre-approval activity into any of its members.

It is also possible to extract another activity pattern: It is shown that the pre-approval needs to be reported prior to the committee's next meeting.

5.5.5 PIPEDA – Accountability Principle

The accountability principle in the Privacy Act stipulates that a designated privacy officer role is responsible and assigned to ensure privacy compliance.

Such a requirement is composed, since it can be translated into several other requirements, such as having the need for a privacy audit process and assigning a privacy officer to that process. Hence, we refer to this example as a composite pattern.

An organisation, for example, may decide to create a privacy process. In the privacy process, there would be sub-processes, such as privacy audit, in addition to privacy governance, reporting, and several others.

Such a translation has to be decided by the governance officers in a given organisation. Because the Act is written in a generic language, the implementation, that is, the creation of the above processes and sub-processes, depends on the evaluation of the judiciary for compliance if a case is raised against the enterprise.

We do not argue the legal aspects in this thesis; however, we can say that, given that the law suggests that the organisation needs to exert reasonable efforts to implement the law, the above process design may fit the legal needs.

5.5.6 PIPEDA – Purpose Identification

The PIPEDA *No collection for unidentified purpose* requirement is high-level and does not discuss any particular process. It only sets the goal that needs to be achieved by the enterprise. Hence, we label this pattern as a declarative pattern.

Such a requirement can be translated into a simpler requirement by placing a conditional checking for availability of the purpose prior to any collection. The translation may suggest a course of action if the collection is denied.

5.6 Conclusion

In this chapter, we have shown a list of patterns often applicable to governance laws. This list accompanies the extraction model that was previously defined in Chapter 2. By using examples, we have shown the possibility of using these patterns to identify requirements.

Chapter 6 will illustrate how the requirements can be properly represented in our Governance Analysis Language (GAL).

Chapter 6

Translating Requirements using Governance Analysis Language

6.0 Introduction

In this chapter, we will discuss in greater detail the method of translating legal requirements into an intermediary language, which can further be translated into a logic analyser model and finally into an Alloy model for validation using the analyser.

In Chapter 5, we have explained how requirements can be extracted and classified into several patterns, such as activities, activity-assignments, delegation, enterprise entities, processes, roles and separation of concerns. We have also discussed high-level patterns that include declarative statements and composite patterns.

In this chapter, we will list a basic set of the Governance Analysis Language that can represent the patterns elaborated in Chapter 5. We provide a translation table between the patterns in Chapter 5 to the GAL; also, we provide an in-depth presentation of the GAL.

6.1 Governance Analysis Language

The GAL can be described as a requirement representation language. The GAL is able to represent the legal and enterprise requirements through the capturing of the requirements using predefined patterns. The patterns represent process and organisational ontologies in addition to the logic operators over the enterprise relations. These relations include resource access rights and delegation modalities. The GAL can be processed by the GATool, which transforms the specification into a formal Alloy model ready for analysis.

In this section, we will provide an introduction to the GAL by the use of examples. The syntax of GAL will be given in section 6.3.

6.1.1 Separation of Concerns example

A rule under SOX is to deny shared access to certain activities or processes. Two examples are: the receipt of goods must be segregated from purchasing and the supplier master data access, and the requisitioning must be segregated from purchasing.

Define Requirements	
Legal	1. The receipt of goods process is separated from the purchasing process and supplier master data access 2. The requisitioning process is separated from the purchasing process
Generation	

Extract	1. The first provision is a separation of concerns pattern 2. Similarly the second provision is a separation of concerns pattern
Represent GAL	1. Separate (Receiving, Purchasing), Separate (Receiving, SupplierMasterAccess)) 2. Separate (Requisitioning, Purchasing)

6.1.2 Right to Delegate Authority example

As provided in SOX and SEC's rules, the Committee may delegate pre-approval authority to one or more of its members. The member to whom such authority is delegated must report, for informational purposes only, any pre-approval decisions to the Committee at its next scheduled meeting.

Generation	
Legal	An audit committee can delegate the activity pre-approval to one or more of its members. The member who receives the delegation must report on approval decisions
Generation	
Extract	1. Role Pattern: This provision suggests the existence of an audit-committee within the enterprise 2. Process Pattern: It is understood from the example that both the access to the Pre-App activity, and to the reporting to committee activities are bundled. Hence, we suggest a single process that includes both activities 3. Activity Pattern: Committee Meeting is an activity that belongs to the main enterprise process 4. Delegation Pattern: The provision also suggests that the audit committee has the right to delegate the process discussed in the previous provision 5. Activity Pattern: The reporting to committee should happen prior to the meeting
Represent GAL	1) Role Pattern: Includes (Issuer, AuditCommittee) 2) Process Pattern: - Contains(App-Report, Pre-App) - Contains(App-Report, Report) 3) Activity pattern:

- Contains(Enterprise, CommitteeMeeting)

4) Delegation Pattern:
   - CanDelegate(Allow, AuditCommittee, AuditCommitteeMember, App-Report)
5) Activity Pattern: To check if the reporting took place prior to meeting:
   - Activity-Trace (Check-Label, App-Report, CommitteeMeeting)

6.1.3 Role and Assignment Patterns example

SOX-Section.2: Audit (3) AUDIT COMMITTEE. The term "audit committee" is defined as a committee (or equivalent body) established by and amongst the board of directors of an issuer for the purpose of overseeing the accounting and financial reporting processes of the issuer and audits of the financial statements of the issuer; and, if no such committee exists with respect to an issuer, the entire board of directors of the issuer.

Define Requirements	
Legal	There are two cases that can be considered here. The issuer has an audit committee or there is none. Option-A 1. Audit Committee can be a subcommittee of the board of directors Option-B 2. If the committee does not exist, then the board of directors acts as the audit committee 3. There can be an equivalent body of the audit committee;

however, this is not specified for now

The audit committee is for the purpose of overseeing the accounting and financial reporting and audits; hence, they have a need to access these processes

Generation

Extract Requirements

Option-A

Assuming that the Board of Directors reports to the Issuer or the main unit in the enterprise, in addition, the AuditCommittee reports to the Board of Directors

- Role Pattern board of directors reports to issuer
- Role Pattern AuditCommittee reports to board of directors

Option-B:

1) Because the Board of Directors will be acting as the audit committee, we then need to equate both roles
   - Definition Pattern, BoardofDirectors has the same label as the AuditCommittee
2) If there is an equivalent body for the audit committee, it would be represented as:
   - Definition Pattern representing the audit committee and the audit pattern
   - The Audit committee needs access to processes accounting, financial reporting and audits processes.

Process pattern providing access to audit committee members to Accounting, FinancialReporting and AuditProcesses

GAL

Option-A

The role pattern suggests building a small hierachy

- Includes(Issuer, BoardofDirectors)
- Includes(BoardofDirectors, AuditCommittee)

Option-B

3) The definition Pattern can be represented as follows
   - EquRole(BoardofDirectors, AuditCommittee)
4) Similarly the definition Pattern can be implemented as follows
   - EquRole(AuditCommittee, OtherBody)
5) The process pattern can be represented as follows
   - Assumes(AuditcommitteeMember,Accounting)
   - Assumes(AuditCommitteeMember,FinancialReporting)
   - Assumes(AuditCommitteeMember,AuditProcess)

6.1.4 Business Process and Activity Patterns example

Canada's financial compliance law (Instruments 52–111) defines the internal control audit report processes and structures using the requirements listed below.

52–111 – Internal control over financial reporting means that a process designed by, or under the supervision of the issuer's chief executive officer and chief financial officer, or persons performing similar functions, and effected by the issuer's board of directors, management and other personnel, to provide reasonable assurance regarding the reliability of financial reporting and the preparation of financial statements for external purposes in accordance with the issuer's GAAP and includes those policies and procedures that:

(a) pertain to the maintenance of records that in reasonable detail accurately and fairly reflect the transactions and dispositions of the assets of the issuer,

(b) provide reasonable assurance that transactions are recorded as necessary to permit preparation of financial statements in accordance with the issuer's GAAP, and that receipts and expenditures of the issuer are being made only in accordance with authorizations of management and directors of the issuer, and

(c) provide reasonable assurance regarding prevention or timely detection of unauthorized acquisition, use or disposition of the issuer's assets that could have a material effect on the annual financial statements or interim financial statements .

Define Requirements

Legal

1. *Internal Control over Financial Process (ICFR) is a process* that is assigned to the Chief Executive Officer and Chief Financial Officer or personnel performing similar functions
2. Establish roles of Board of Directors, management
3. ICFR is affected by Board of Directors, Management and others

Board of Directors, Management and others needs to have access to the ICFR process

4. The ICFR process needs to:
    a. Provide reasonable assurance of financial reporting
    b. Prepare financial statements
    c. Check detailed accuracy of records
    d. Check GAAP compliance
    e. Check if expenditures have authorizations
    f. Provide an activity to check for unauthorized acquisition
    g. Provide an activity to check for disposition of assets.

Generation

Extract

1. Process Pattern
    a. Establish the ICFR process under the issuers process
2. Activity Pattern
    a. Assign the ICFR to the CFO and the CEO
3. Role Pattern: Issuer includes Board of Directors and management role
4. Activity Pattern: The ICFR process includes the following activities:
    a. Provide reasonable assurance of financial reporting
    b. Preparation of financial statements
    c. Check detailed accuracy of records
    d. Check GAAP compliance
    e. Check if expenditures have authorizations
    f. Provide an activity to check for unauthorized acquisition
    g. Provide an activity to check for disposition of assets.

6.2 Representation Process

To derive the GAL statements from the legal requirements, a Governance Officer needs to follow our systematic process of pattern extraction as represented in the previous chapter. Once the patterns are extracted, the GO needs to provide the equivalent GAL statements based on the following translation table.

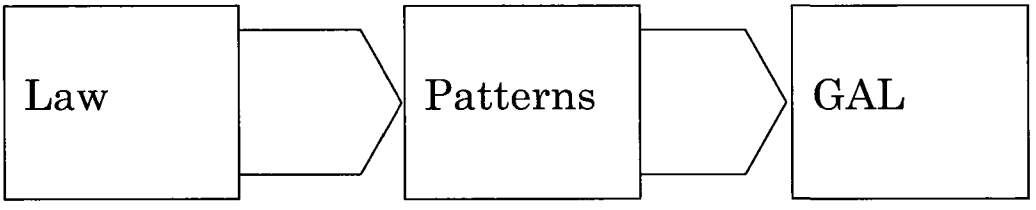


Figure 6.1: Representation process

Figure 6.1 illustrates that the translation process starts with extracting requirements from the law, the extraction process uses the patterns categorisation. The pattern to GAL mapping table is used to assist in translating from patterns to GAL.

Each of the patterns can be replaced by the GAL statements in accordance with the following table.

Table 6.1: Pattern to the GAL translation

Activity Pattern	
Declaration	Contains (Process, Activity)
	EquActivity(Activity-1, Activity-2)
Relation	Next (Activity, Activity)

	Contains (Process, Activity)	
	AssignedTo(Role, Activity)	
Checks	Process-Activity	(Check-Label, Process, Activity,)
	Activity- Predecessor	(Check-Label, Activity-1, Activity-2)
	Activity-Trace	(Check-Label, Activity-1, Activity-2)
	Activity-Process-Pred	(Check-Label, Process-1, Activity-1, Activity-2)
	Activity-Process-Trace	(Check-Label, Process-1, Activity-1, Activity-2)
	Instance	(Check-Label, Activity)
	checkAssignedTo	(Check-Label, Role, Activity)

Definition Pattern

Declaration	EquActivity	(Activity-1, Activity-2)
	EquProcess	(Process-1, Process-2)
	EquRole	(Role-1, Role-2)
Relation	N/A	
Check	N/A	

Process Pattern

Declaration	ComposedOf	(CompositeProcess, [Atomic-Process])
	Process (Process-1, Process-2)	
Relation	Contains (Process, Activity)	
	Assumes (User, Process)	
Policy	CanAssume	((Allow | Deny), Process, User)
	Separate	((Process-1, Process-2), (Process-2,Process-3))
Checks	Process-Parent	(Check-Label, Process-1, Child-1)
	Process-Activity	(Check-Label, Process, Activity,)
	Activity-Process-Pred	(Check-Label, Process-1, Activity-1, Activity-2)
	Activity-Process-Trace	(Check-Label, Process-1, Activity-1,Activity-2)

	Instance	(Check-Label,Process)
	checkAssumes	(Check-Label,User, Process)

Department/Role Pattern

Declaration	Includes (Department/Role, Department/Role)	
Relation	AssignedTo(Role, Activity)	
	Acts(Role, User)	
Policy	CanAct	((Allow | Deny), User, Role-1, [Role-2])
	CanAssignTo	((Allow | Deny), Activity, Role)
	CanDelegate	((Allow | Deny), Role, Role, [Activity|Process])
Checks	Dep-Parent	(Check-Label, Dept-Parent-1 , Child-1)
	Instance	(Check-Label, Department/Role)
	checkActs	(Check-Label, Role, User)
	checkAssignedTo	(Check-Label, Role, Activity)
	checkDelegate	(Check-Label, Role, Role)

Assignment Pattern

Declaration	Assumes	(User, Process)
	AssignedTo(Role, Activity)	
	Acts	(Role, User)
Classes	User, Process, Role, Activity	
Policy	CanAssume	((Allow | Deny), Process, User)
	CanAssignTo	((Allow | Deny), Activity,Role)
	CanAct	((Allow | Deny), User, Role)

Check	checkActs	(Check-Label, Role, User)
	checkAssignedTo	(Check-Label, Role, Activity)
	checkAssumes	(Check-Label, User, Process)

Delegation Right Pattern

Declaration	CanDelegate ((Allow | Deny), Role, Role, [Activity|Process])
Classes	Role
Policy	N/A
Check	checkDelegate ((Allow | Deny), Role, Role)

Separation of Concerns Pattern

Declaration	Separate((Process-1, Process-2), (Process-2, Process-3))
Classes	Process
Policy	N/A
Check	N/A

Group Assert

Declaration	Group-Assert	(Check-Labelx, Check-Labelxx, ...)
Classes	All	
Policy	All	
Check	All	

6.3 GAL statement reference

We use first-order logic represented by a set theory, which provides the basic concepts of atoms and relationships. This logic representation uses parentheses, a finite set of objects (limited to the cardinality of the enterprise model in question), existential and universal quantifiers, transitive closures and standard propositional logic connectives such as negation, conjunction, disjunction, implication and equivalence.

Constant symbols are used for denoting enterprise entities such as Privacy or Chief Information Officers. In addition, we will make references to potential processes and their steps as entities.

We find it helpful to restate our definition of the set of requirements as being:

$\Phi = \{ P, A, R, U, X, D, S, RA, RR, RU, PA, UP, PP\}$

Now we will present the basic GAL language statements representing the legal requirements model. For each of the statements, we will introduce semantics, syntax, an example and a formal representation.

Table 6.2: Formal model examples

Set	Set of	Example
P	Processes	(Finance)
A	Activities	(ReviewApplication)
R	Roles	(Director)
U	Users	(John Doe)
X	Process Separated from Process	(Finance, Audit)
D	Privacy Officer can delegate to chief security officer	(Privacy Officer, Security Officer, DocAccess)
S	Activity Sequences	(Accept, Sign)
RA	Role Activity assignment	(Director, ReviewApplication)
RR	Role contains Role relation	(Human Resources, Recruitment)
RU	Role to User	(FinanceAdminAssistant,Jane)
PA	Process to Activity	(Privacy Audit, ReviewPurposeList)
UP	User Process assignment	(John Doe, Finance)
PP	Process contains Process relation	(Privacy, Privacy Audit)

Semantics attributes describe the meaning of each rule, while syntax attributes describe their form. We will use first-order logic for the formal representation. An example is given from our implementation.

ComposedOf

Composed Of	
Syntax	ComposedOf (CompositeProcess, [Atomic-Process])
Semantics	This statement suggests a process hierarchy. A composed process can have one or more sub-processes
Logic	(CompositeProcess, AtomicProcess-a) ∈ PP
Example	The audit process includes financial audit and privacy audit sub-processes ComposedOf(Audit, Financial Audit) ComposedOf(Audit, Privacy Audit)

Contains

Contains	
Syntax	Contains (Process,Activity)
Semantics	A process contains a certain activity.
Logic	(Process-1, Activity-1) ∈ PA
Example	Financial audit process includes the signing of financial compliance certification in addition to issuing a statement of account to the regulatory board Contains(FinancialAudit, signingOfFinancialCompliance) Contains(FinancialAudit, IssuingStatementtoRegBoard)

Next

Next	
Syntax	Next (Activity, Activity)
Semantics	Activities are placed in a sequence.
Logic	(Activity-1, Activity-2) ∈ S
Example	The signingOfFinancialCompliance requires board approval Next(BoardApproval, SigningOfFinancialCompliance)

Includes

Includes	
Syntax	Includes (Department/Role, Department/Role)
Semantics	A department may contain one or more departments.
Logic	(Dept-1,Dept-2) ∈ RR
Example	The HR Department contains the bragaining unit team Includes (HR, BargainingUnitTeam)

EquActivity

EquActivity	
Syntax	EquActivity(Activity-1,Activity-2)
Semantics	Activity-1 can be represented by Activity-2
Example	A collection exemption can be sickness, long-term disability or minor. EquActivity (ExemptCollection, sicknessExemption) EquActivity (ExemptCollection, long-termExemption) EquActivity (ExemptCollection, minorExemption)

EquProcess

EquProcess	
Syntax	EquProcess (Process-1, Process-2)
Semantics	Process-1 can be represented by Process-2
Example	*Privacy Technology Audit Process can be replaced by* Finance Technology Audit Process EquProcess(PrivacyTechAuditP, FinanceTechAuditP)

EquRole

EquRole	
Syntax	EquRole(Role-1, Role-2)
Semantics	Role-1 can be represented by Role-2
Example	Financial admin assistants can represent admin assistants EquRole(FinancialAdminAssistants,AdminAssistants)

Assumes

Assumes	
Syntax	Assumes (User, Process)
Semantics	A user assumes a process. In other words, he has access to the processes activities.
Logic	(User-1, Process-1) $\in$ UP
Example	A privacy officer assumes the privacy process Assumes(PrivacyOfficer,PrivacyProcess)

AssignedTo

AssignedTo	
Syntax	AssignedTo(Role, Activity)
Semantics	An activity is assigned to a role, *i.e.* the activity is a part of the role's credentials.
Logic	(Role-1, Activity-1) ∈ RA
Example	A chief financial officer is tasked with signing financial statements AssignedTo(CFO, signingFinancialStatements)

Acts

Acts	
Syntax	Acts(Role, User)
Semantics	A user acts in a particular role. When acting in a role, a user gains access Role`s activities
Logic	(Role-1, User-1) ∈ RU
Example	Lara Smith works as a financialAnalyst Acts(FinancialAnalyst, LaraSmith)

Separate

Separate	
Syntax	Separate((Process-1, Process-2), (Process-2,Process-3))
Semantics	Users who have access to Process-1 cannot have access to Process-2
Logic	(Process-1, Process-2) ∈ X

Example	Separate Financial Audits from Privacy Audits Separate(FinancialAudit, PrivacyAudit)

Delegate

Delegate	
Syntax	Delegate (User|Role, User|Role, [Activity| Process])
Semantics	A Role delegates a user or an activity to another role
Logic	(User-1, User-2, Activity-1) $\in$ D
Example	A Financial Officer (Ali) delegates his ability to sign privacy compliance statements to a Chief Executive Officer (John) Delegate(CFO| Ali, CEO|John, signComplianceDoc)

CanAssume

CanAssume	
Syntax	CanAssume ((Allow | Deny), Process, User)
Semantics	A user can assume or be prevented from assuming a process
Logic	(User-1, Process-1) $\notin$ UP, user-1 cannot assume Process-1
Example	A finance admin assistant cannot assume FinancialReviewProcess canAssume(Deny, FinanceAdminAssistant, FinancialReviewProcess)

CanAssignTo

CanAssignTo	
Syntax	CanAssignTo ((Allow | Deny), Activity,Role)
Semantics	An activity can be assigned to a role

Logic	(Role-1, Activity-1) ∈ RA
Example	Receiving returned mail cannot be assigned to mail dispatchers canAssignTo(Deny, ReceivingReturnedMail, MailDispatchers)

CanDelegate

CanDelegate	
Syntax	CanDelegate ((Allow | Deny), Role, Role,[Activity| Process])
Semantics	A Role can be delegated to another role
Logic	(Role-1, Role-2, Activity-1) ∈ D
Example	A Financial Officer can delegate his ability to sign privacy compliance statements to a Chief Executive Officer canDelegate(Allow, CFO, CEO. signComplianceDoc)

CanAct

CanAct	
Syntax	CanAct ((Allow | Deny),User, Role)
Semantics	A user can either act or not in the a particular role(s)
Logic	(Role-1, User-1) ∉ RU
Example	JaneSmith can act as a manager of finance canAct(Allow, JaneSmith, FinanceManger)

Process-Parent

Process-Parent		
Syntax	Process-Parent	(Check-Label,Process, Child)
Semantics	Validate if a process has the specified child processes.	
Logic-Test	(Process, Child) ∈ PP	
Example	Check if privacy process has an audit sub-process. Process-parent(Provision-1.0, Privacy, PrivacyAudit)	

Process-Activity

Process-Activity		
Syntax	Process-Activity	(Check-Label, Process, Activity,)
Semantics	Checks if a process includes the specified activities. The result is boolean.	
Logic	(Process, Activity) ∈ PA	
Example	Check if the privacy audit process contains a purpose list collection activity. Process-Activity(Provision-1.1,ve Privacy, PurposeListCollection)	

Dep-Parent

Dep Parent Pattern		
Syntax	Dep-Parent	(Check-Label, Dept-Parent , Child)
Semantics	Checks if a Department/Role has child dept/roles	

Logic	(Dept-Parent,Child) ∈ RR
Example	Check if the finance department has an audit division Dep-Parent(Provision-1.2,Finance,Audit)

Activity-Predecessor

Activity-Predecessor

Syntax	Activity- Predecessor (Check-Label, Activity-1 ,Activity-2)
Semantics	Check if the immediate predecessor of an activity is as specified
Logic	(Activity-1,Activity2) ∈ S
Example	Check if data collection was preceded by a purpose declaration Activity-Predecessor (Provision-1.3, PurposeDeclaration, DataCollection)

Activity-Trace

Activity-Trace

Syntax	Activity-Trace (Check-Label, Activity-1, Activity-2)
Semantics	Check if an activity is in any of the ancestors of another
Logic	Activity-2 ∈ Activity-1.^next Where ^ indicates the closure set
Example	Forwarding of Personal Information should be preceded by an agreement with partners Activity-Trace(Provision-1.4, Agreement, forwardingPI)

Activity-Process-Pred

Activity-Process-Pred	
Syntax	Activity-Process-Pred (Check-Label, Process, Activity-1 ,Activity-2)
Semantics	Check if an activity has a predecessor within the same process
Logic	{(Process-1,Activity-1), (Process-1, Activity-2)} ∈ PA (Activity-1, Activity2) ∈ S
Example	Client consent is requested prior to checking credit history in credit approval process. Activity-Process-Pred(Provision-1.5, CreditApproval, CollectConsent, CreditCheck)

Activity-Process-Trace

Activity-Process-Trace	
Syntax	Activity-Process-Trace(Check-Label, Process, Activity-1,Activity-2)
Semantics	There is a path from one activity to another within the same process
Logic	{(Process-1, Activity-1), (Process-1, Activity-2)} ∈ PA Activity-2 ∈ Activity-1.^next
Example	Notification of account cancellation is sent to client prior to data disposal in the cancellation process Activity-Process-Trace(Provision-1.6, datadisposal, CancellationProcess)

checkInstance

checkInstance	
Syntax	CheckInstance(Check-Label, User | Process | Department/Role | Activity)
Semantics	Ensures that a specific class instance exists
Logic	$\exists$ (User | Process | Department/Role | Activity) $\in \Phi$
Example	Ensure there is an privacy audit process called Paudit CheckInstance(Paudit)

checkAssumes

checkAssumes		
Syntax	checkAssumes	(Check-Label, User, Process)
Semantics	Check if a user assumes a process	
Logic	(User, Process) $\in$ UP	
Example	Check if John Doe has access to auditprocess checkAssumes(AuditProcess, JohnDoe)	

checkAssignedTo

checkAssignedTo		
Syntax	checkAssignedTo	(Check-Label, Role, Activity)
Semantics	Check if an activity is assigned to a role, *i.e.* the activity is a part of the role's credentials.	
Logic	(Role, Activity) $\in$ RA	

Example	Check if signing of financial statements is assigned to financial officer checkAssignedTo(CFO, signFinancialStatement)

checkActs

checkActs		
Syntax	checkActs	(Check-Label, Role, User)
Semantics	A user acts in a particular role and gains access to his activities	
Logic	(Role, User) ∈ RU	
Example	Test if SarahSmith is acting as a business analyst checkActs(SarahSmith, BA)	

checkDelegate

checkDelegate		
Syntax	checkDelegate	(Check-Label, Role, Role)
Semantics	Check if it is possible to delegate one role to another	
Logic	(Role-1, Role-2) ∈ D	
Example	Check if it is possible to delegate the role CPO to CFO checkDelegate	(Check-Label, CPO, CFO)

GroupAssert

Group Assert		
Syntax	Group-Assert	(Check-Labelx, Check-Labelxx, ...)
Semantics	Checks multiple statements at once. The Check-labels are given.	
Logic	N/A	

ALL

Syntax	ALL
Semantics	When passed as a parameter to a predicate it indicates that the formula is true for all members of the class or relation

SOME

Syntax	SOME
Semantics	When passed as a parameter to a predicate it indicates that the expression is true for some members of the class or relation

NONE

Syntax	NONE
Semantics	When passed as a parameter to a predicate it indicates that the predicate is true for none of the members of the class or relation

6.4 More on GAL

In this section, we provide several groupings of the GAL language for reference.

6.4.1 Alphabetical List

In this section, we present a simple alphabetical list of GAL statements for reference.

1. Activity- Predecessor (Check-Label, Activity-1 [,Activity-2])
2. Activity-Process-Pred (Check-Label, Process-1, Activity-1, Activity-2)
3. Activity-Process-Trace(Check-Label, Process-1, Activity-1, Activity-2)
4. Activity-Trace (Check-Label, Activity-1, Activity-2)
5. Acts (Role, User)
6. AssignedTo (Role, Activity)
7. Assumes (User, Process)
8. CanAct ((Allow | Deny), User, Role)
9. CanAssignTo ((Allow | Deny), Activity, Role)
10. CanAssume ((Allow | Deny), Process, User)
11. CanDelegate ((Allow | Deny), Role, Role, [Activity | Process])
12. checkActs (Check-Label, Role, User)
13. checkAssignedTo (Check-Label, Role, Activity)
14. checkAssumes (Check-Label, User, Process)
15. checkDelegate (Check-Label, Role, Role)

16. checkInstance (Check-Label, User | Process | Department/Role | Activity)
17. ComposedOf (CompositeProcess, [Atomic-Process])
18. Contains (Process, Activity)
19. Delegate (Role | User, Role | User, [Activity | Process])
20. Dep-Parent (Check-Label, Dept-Parent, Child)
21. EquActivity (Activity-1, Activity-2)
22. EquProcess (Process-1, Process-2)
23. EquRole (Role-1, Role-2)
24. Group-Assert (Check-Labelx, Check-Labelxx, ...)
25. Includes (Department/Role, [Department/Role])
26. Next (Activity, Activity)
27. Process-Activity (Check-Label, Process, Activity)
28. Process-Parent (Check-Label, Process, Child)
29. Separate ((Process-1, Process-2)[(Process-2, Process-3)])

6.4.2 Grouped by function

In this section, we provide a listing of the GAL statements grouped by function.

Construction

1. ComposedOf (CompositeProcess, [Atomic-Process])
2. Contains (Process, Activity)
3. Includes (Department/Role, [Department/Role])

Equivalence

1. EquActivity (Activity-1, Activity-2)
2. EquProcess (Process-1, Process-2)
3. EquRole (Role-1, Role-2)

Relational

1. Acts (Role, User)
2. AssignedTo (Role, Activity)
3. Assumes (User, Process)
4. Delegate (Role|User, Role|User, [Activity|Process])
5. Next (Activity, Activity)
6. Separate ((Process-1, Process-2)[(Process-2, Process-3)])

Assignments

1. CanAct ((Allow | Deny), User, Role)
2. CanAssignTo ((Allow | Deny), Activity, Role)
3. CanAssume ((Allow | Deny), Process, User)
4. CanDelegate ((Allow | Deny), Role, Role, [Activity|Process])

Checks

1. Activity- Predecessor (Check-Label, Activity-1 [,,Activity-2])
2. Activity-Process-Pred (Check-Label, Process-1, Activity-1 ,Activity-2)
3. Activity-Process-Trace(Check-Label, Process-1, Activity-1,Activity-2)

4. Activity-Trace (Check-Label, Activity-1, Activity-2)
5. checkActs (Check-Label, Role, User)
6. checkAssignedTo (Check-Label, Role, Activity)
7. checkAssumes (Check-Label, User, Process)
8. checkDelegate (Check-Label, Role,Role)
9. checkInstance (Check-Label, User | Process | Department/Role| Activity)
10. Dep-Parent (Check-Label, Dept-Parent, Child)
11. Group-Assert (Check-Labelx, Check-Labelxx, ...)
12. Process-Activity (Check-Label, Process, Activity)
13. Process-Parent (Check-Label, Process, Child)

6.5 Conclusion

In this chapter, we have shown how the results of the legal text analysis can be translated into a governance language from there into logic. We have shown the logic mapping of each of the basic instructions, in addition to some composite patterns. Several examples of translation were given.

The next chapters will show examples of the application of this method in privacy and financial governance domains.

Chapter 7

Governance Analysis Method for Privacy – Case Study

7.0 Introduction

This chapter offers a proof of concept for the governance analysis method. We present several enterprise requirements to be validated against PIPEDA. In the previous two chapters our examples were limited to showing extraction and translation of requirements. However, this chapter will also show extraction and translation into GAL statements as well as equivalent logic representations. For each solution, a visualised presentation showing formatted GAT output using the Alloy engine will be presented. The chapter will be concluded with a discussion on lessons learned.

In section 7.1 we will present a background reminder of the PIPEDA. Our case study will show four examples taken from the PIPEDA that will be summarized in section 7.2. The case study details will be presented in sections 7.3 and 7.4.

7.1 Background and Case Summary

We find it useful to provide a short overview of the Privacy Act in the first part of this section. The second part will provide a high level summary of the examples and results as well as any conflicts found.

7.1.1 Background

In Canada, the Privacy Act governs how private-sector organizations collect, use, and disclose personal information in the course of commercial transactions. In addition, it contains various provisions to define governance requirements for the exchange of personal information.

PIPEDA gives individuals the right to:

1. know why organisations collect, use or disclose their personal information;
2. expect organisations to collect, use or disclose their personal information reasonably and appropriately, and not to use this information for any purpose other than that to which they have consented;
3. know who in an organisation is responsible for protecting their personal information;
4. expect that organisations will protect their personal information by taking appropriate security measures;
5. expect that the personal information organisations hold about them is accurate, complete and up-to-date;
6. obtain access to their personal information and ask for corrections, if necessary;
7. complain about how organisations may handle their personal information if they feel their privacy rights have not been respected.

The next subsection we will present a summary of the problem statement in each of the examples in the case study and our respective findings after applying the GAM.

7.1.2 Summary of Case Study Examples

In each of the examples, we will present a legal and enterprise requirements as well as the goals of the governance officer. Our method will be applied to each of the examples. Furthermore, we will provide a summary of the findings. The detailed analysis of the case study can be found in the next section.

Example 1: Accountability Violation

We will use the accountability principle from PIPEDA for this illustration.

Legal: Provision 4.1.0 requires that organisations have a privacy officer role with named individuals. Provision 4.1.1 states that the privacy role can be delegated. Provision 4.1.2 stipulates that the identity of the privacy officer needs to be made available through a process.

Enterprise: Consider now a specific organisation that claims to implement PIPEDA. The organisation has established a privacy officer role, but the individual in this role has delegated his duties to another individual for reasons of absence.

GO: The governance officer acting on behalf of the Privacy Commissioner's audit is tasked with testing Provision 4.1.x.

Findings: The organisation is deemed compliant with provisions 4.1.0 and 4.1.1.

First, this is true because the organisation has established a privacy officer role as required by provision 4.1.

Second, the individual in the privacy officer role has delegated his duties to another individual for reasons of absence, is permitted by provision 4.1.2. The creation of a privacy officer role and the delegation of duties attached to this role for valid reasons are both permitted activities under PIPEDA. However, one possible scenario in which the organisation could be found to be in violation of provision 4.1.2 would be if the privacy process provides the name of the original privacy officer rather than the name of his delegate.

Example 2: Access Contradiction

Legal: One of the requirements of PIPEDA is to *require a purpose justifying any access to personal information.*

Enterprise: Consider a situation in which employees work in the privacy division of an enterprise. These employees deal with enterprise governance aspects, but do not have a purpose for accessing personal information and thus, do not enjoy the ability to access any personal information.

GO: The GO needs to validate that there are no violations to the requirements

Findings: It may be possible that the privacy department and the finance department of this enterprise share an administrative assistant. The assistant's role in the finance department may allow access to personal information of clients, whereas her role in the privacy division prevents her the same access.

Using our method this interaction can be detected through a model consistency check.

Example 3: Ontology Sanity

The privacy law has several provisions for specifying structural aspects and processes of organisations. As trivial as it might first appear, such a check can be made complex by ontology relations.

Legal: Consider a legal requirement of having a privacy process and a privacy audit sub-process in an enterprise. Another PIPEDA rule may ask that *All data collection needs to be preceded by a purpose declaration.* A compliance rule may suggest that *an agreement to respect privacy needs to be signed with third party prior to sending the clients' personal information.*

Enterprise: At the same time, an enterprise requirement may request that the same audit sub-process be part of their corporate governance audit owned by the financial division.

GO: In this example, the governance officer's role is to check the various provisions and their validity.

Findings: All sub-examples were found to be compliant.

Example 4: Limiting Use, Disclosure and Retention Violation

Legal: The fifth principle of PIPEDA expressed in provision 4.5.3 suggests that *personal information shall be retained only as long as the purpose of collection is not reached.*

Enterprise: Consider a process collecting personal information for a specific purpose. This process depends on a sister process in order to be fulfilled, resulting in both processes sharing the collected personal information.

GO: The governance officer's job is to validate that no information is retained once its purpose is achieved.

Findings: The organisation could be found to be in violation of provision 4.5.3. A scenario check may find that once the collecting process has achieved its purpose, the shared (forwarded) data continues to be retained by the sister process, thus resulting in the violation of provision 4.5.3.

7.2 Case Study : Enterprise Requirements

This section will present the enterprise requirements, which will be used through out the various examples of the case study presented in section 7.3.

7.2.1 Enterprise Process

Privacy Processes

For the purpose of the PIPEDA case study, we will use a sample enterprise defined through the diagrams and associated text indicated below

Figure 7.1 presents two privacy processes, the Privacy and Privacy Officer Identity (POIdentity).

The privacy process is intended to deliver the privacy policy of the enterprise upon client request. The POIdentity sub-process is intended to provide a client with the identity of the privacy officer. It has three activities and starts with a client's request, continues with the delivery of the privacy process identity and concludes with the logging of the request by the process.

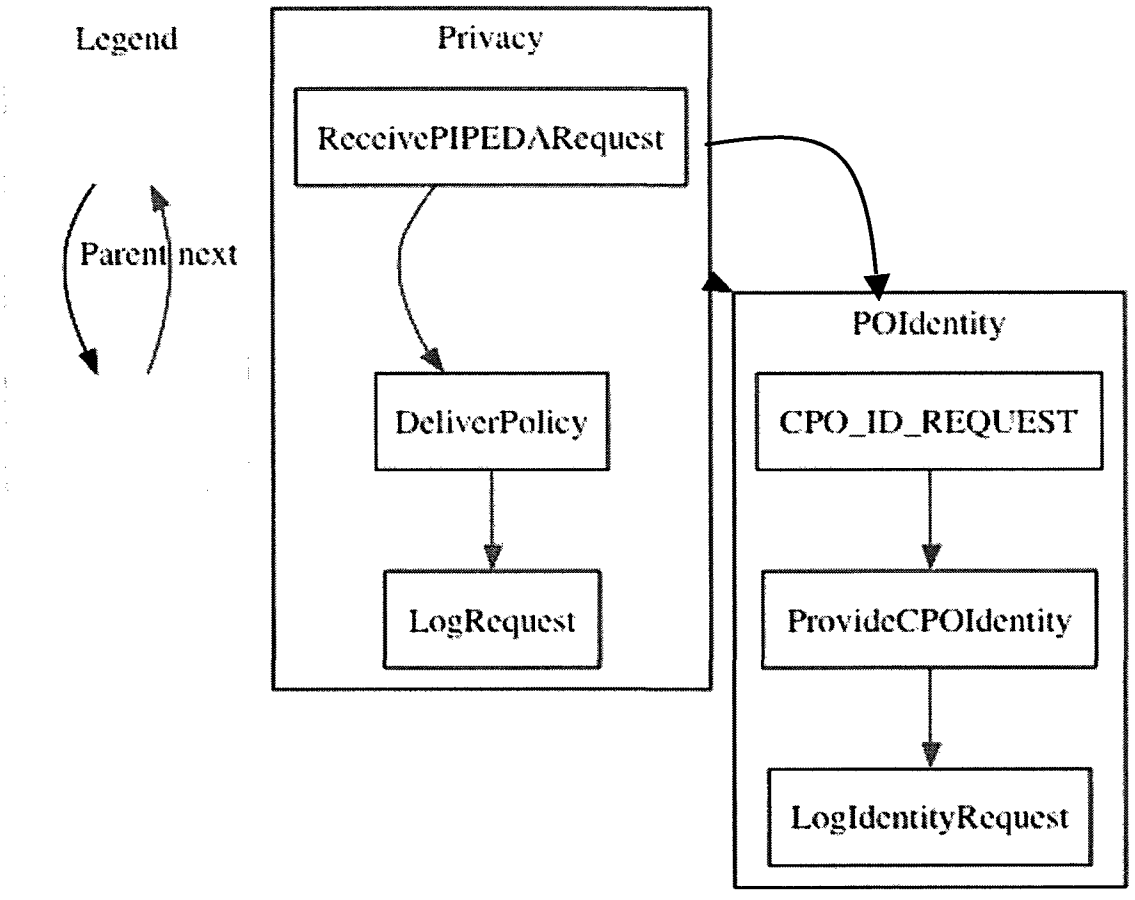


Figure 7.1: Enterprise Privacy Processes

GAL Representation of Enterprise Processes:

Process Pattern

```
ComposedOf(Enterprise,Privacy)

ComposedOf(Pirvacy,POIdentity)

Contains(POIDentity, CPO_ID_Request)
```

Contains(POIDentity,ProvideCPOIdentity)

Contains(POIDentity,LogIdentityRequest)

Contains(Privacy, ReceivePIPEDARequest)

Contains(Privacy, DeliverPolicy,LogRequest)

Activity Pattern

Next(DeliverPolicy ,LogRequest)

Next(ReceivePIPEDARequest, DeliverPolicy)

Next(CPO_ID_Request,ProvideCPOIdentity)

Next(ProvideCPOIdentity,LogIdentityRequest)

Loans and Order Management Processes

Figure 7.2 illustrates the loans and order management processes. The loans process includes the Publish Application, Receive Filled Application, Write Application, Justify Received Application, Client Consent, Legal Reason Exception, Thank Client, and Dispose Data Activities. The order management process includes the Read Application, Validate Information, and Save Information activities. The flows are provided in Figure 7.2.

GAL Representation of Loans and Order Management Process:

Process Pattern

ComposedOf(Enterprise, Loans)

ComposedOf(Enterprise,OrderMgt)

Contains (Loans, PublishApplication)

Contains (Loans, ReceiveFilledApp)

Contains (Loans, Wapplication)

Contains (Loans, JReceiveFilledApp)

Contains (Loans, ConsentClient)

Contains (Loans, LegalReasonException)

Contains (Loans,ThankClient,DisposeData)

Contains(OrderMgt, ReadApplication)

Contains(OrderMgt, ValidateInfo)

Contains(OrderMgt,SaveInfo)

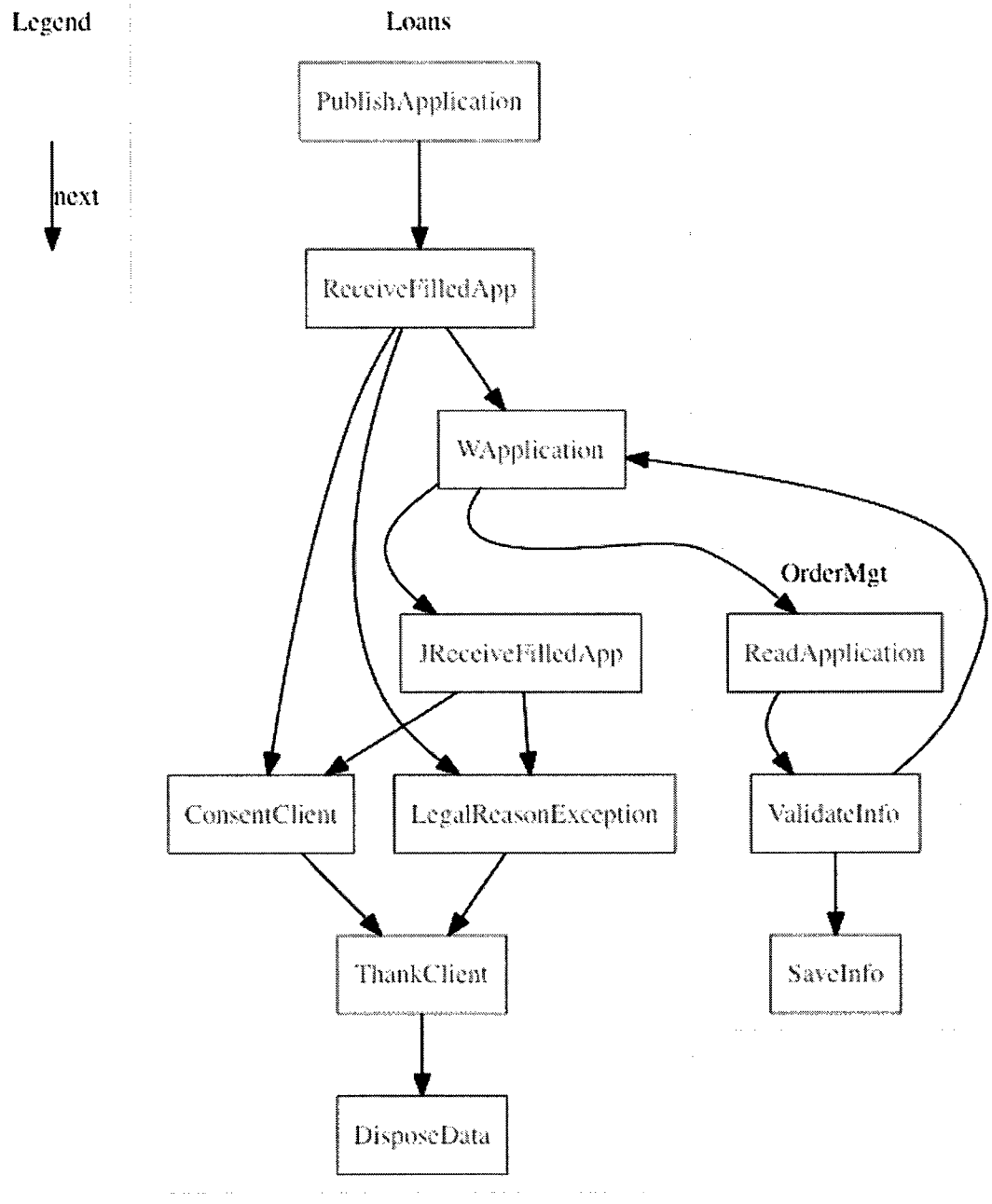


Figure 7.2: Loans and Order Management Processes

Activity Pattern

Next (ValidateInfo,SaveInfo)

Next(ReadApplication, ValidateInfo)

Next(Wapplication, JreceivedApp)

Next(JReceivedApp,ConsentClient)

Next(JReceivedApp,LegalReasonException) Next(ThankClient,DisposeData)

Next(PublishApplication, ReceiveFilledApp)

Next(ReceiveFilledApp,Wapplication)

Next(ValidateInfo,WApplication)

Next(WApplication,ReadApplication)

7.2.2 Enterprise Organisational Structure

Figure 7.3 illustrates the enterprise structure. The Chief Executive Officer (CEO) is the most senior role in the enterprise. Financial, privacy, and technology officers report to the CEO. The Director for Audit (DirectorAudit), Director for Commercial Clients (DirectorCommercial) and the Director for Retail (DirectorRetail) all report to the Vice President of Banking Operations, who in return reports directly to the CEO.

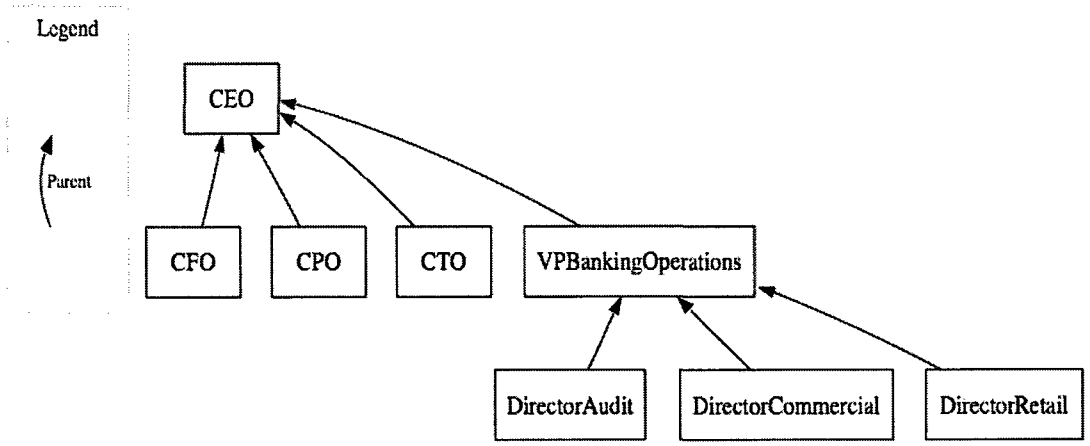


Figure 7.3: Organisational Structure

GAL Representation of Organisational Structure:

Role Pattern

```
Includes(CEO,CFO) Includes(CEO,CPO) Includes(CEO, CTO)

Includes(CEO,VPBankingOperations)

Includes(VPBankingOperations,DirectorAudit)

Includes(VPBankingOperations,DirectorCommercial)

Includes(VPBankingOperations, DirectorRetail)
```

7.2.3 Enterprise Relations

Figure 7.4 shows several assignment and process patterns. Jane assumes the privacy process and holds the role of the Chief Privacy Officer (CPO). The

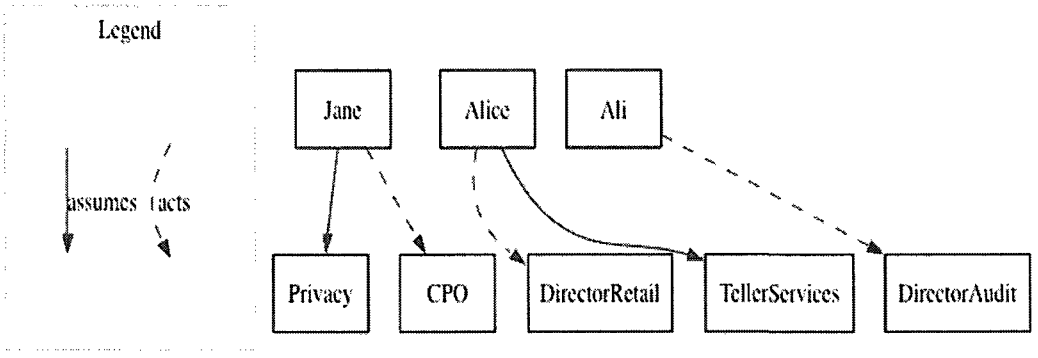


Figure 7.4: Enterprise Relations

Figure in 7.4 illustrates the defined assignments as per the enterprise configuration. For example: Alice acts as of the Director of Retail (DirectorRetail) and works for the Teller Services process (TellerServices). Ali acts as the Director of Audit (DirectorAudit).

Gal Representation of Enterprise Relations:

Process Pattern:

```
Assumes(Jane,Privacy)
Assumes(Alice,TellerServices)
```

Role Pattern

Acts(Jane,CPO)
Acts(Alice,DirectorRetail)
Acts(Ali,DirectorAudit)

7.3 Case Study : Compliance Analysis

In this sub-section, we will apply the same four examples previously summarized in section 7.2.

7.3.1 Example 1: Accountability Violation

The first activity in the process is to define the requirements and prepare for extracting patterns.

Define Requirements	
Legal	1. Provision 4.1 requires that organisations have a privacy officer role with named individuals. 2. Provision 4.1.1 states that the privacy role can be delegated. 3. Provision 4.1.2 stipulates that the identity of the privacy officer needs to be made available through a process.
Enterprise	4. This organisation has established a privacy officer. 5. The individual has delegated his or her role while in absence.
GO	6. Check if Alice is the PO

Following the method we extract a role and an assignment pattern from provision 4.10. Furthermore, from 4.1.1 we extract a delegation pattern. Finally from 4.12 we extract a process pattern.

Below, we will explain each pattern and the corresponding legal requirement.

Generation	
Extract	**Patterns** 1) Provision 4.1.0: • Role Pattern: a privacy officer role (PO) • Assignment Pattern: Check Privacy Officer Role is not empty
Legal	2) Right to Delegate Pattern: Privacy Officer Role can be delegated 3) Process Pattern: The identity of the Privacy Officer needs to be made available through a process.
Enterprise	• Assignment: Assign an individual to PO role • Delegation: Delegate the PO role to another individual
GO	• Assignment Pattern: Check if Jane is assigned to PO role • Activity Pattern: Check if an identity request activity is followed by a delivery activity

Once the patterns are extracted, the generation process proceeds to the translation activity, which we show next.

Generation	
Translate to GAL	1) Provision 4.1.0: - Includes(CEO, PO) - checkActs(PO,ANY)
Legal	2) Right to Delegate Pattern: - canDelegate(Allow , PO, ANY) 3) Process Pattern: - ComposedOf(Enterprise,POIdentity) - Contains(POIdentity, CPOIDRequest) - Contains(POIdentity, ProvideCPOIdentity) - Next(CPO_ID_Request,PRovideCPOIdentity)

Enterprise	4) Acts (PO, Jane) 5) Delegate (Jane, Alice)
Represent in Logic	Φ = { P, A, R, U, X,S,D,RA,PA,UP,RR,PP} 1. Provision 4.1.0: a. (CEO, PO) $\in$ RR b. $\exists$ X, (PO,X) $\in$ RR 2. Right to Delegate Pattern: a. $\forall$ r $\in$ R,$\forall$ a $\in$ A (PO, r,a) $\in$ D 3. Process Pattern: a. (Enterprise,POIdentity) $\in$ PP b. (POIdentity, CPOIDRequest) $\in$ PA c. (POIdentity, ProvideCPOIdentity) $\in$PA d. (CPO_ID_Request,PRovideCPOIdentity) $\in$ S

The second table illustrates extracted statements and assertions required to arrive at the results. The logic representation is presented for informational

Consistency Validation	
Ontology Check	checkActs (Provision-4.1.0, PO, Alice) Process-Activity (Provision-4.1.1, CPO_IdentityRequest, ProvideCPOIdentity)
Represent in Logic	Φ = { P, A, R, U, X,S,D,RA,PA,UP,RR,PP} • (PO, Alice) $\in$ RR • (CPOIDRequest, ProvideCPOIdentity) $\in$ PA
Model Consistency Check	Con (Φ) the model was found to be logically consistent
Scenario Check	N/A

purposes only.

7.3.2 Example 2: Access Right Interaction

As in this example, we apply the GAM to an access right contradiction example.

Define Requirements	
Legal	6) Users in the Privacy division do not have access to PI.
Enterprise	7) The enterprise includes a division for finance and one for privacy 8) Admin assistants in the Finance division are allowed access to review PI. 9) Admin-assistant-1 belongs to both Finance and Privacy divisions
Generation	
Extract	-Privacy division is not able to access activity Access Personal Information. -Admin assistants in Finance have access to PI
Represent GAL	10) Assignment Pattern: - CanAssignTo(Deny,AccessPI,Privacy-Div) 11) Role Pattern: - Includes(Eneterprise,Fin-Div) - Includes(Eneterprise, Priv-Division) - Includes(Fin-Div, AdminAssistants) 12) Assignment Pattern: - CanAssignTo(Allow, AccessPI,FinAdminAssistants) - Acts(Admin-Assistant-1,Privacy-Div, FinAdminAssistants)
GO	Check the model for inconsistencies
Represent in Logic	Φ = { P, A, R, U, X,S,D,RA,PA,UP,RR,PP} 1) Assignment Pattern: - {(Priv-Div,AccessPI)} $\not\subset$ RA 2) Role Pattern: - (Eneterprise,Fin-Div) $\in$ RR - (Eneterprise, Priv-Division)$\in$ RR - (Fin-Div, AdminAssistants)$\in$ RR 3) Assignment Pattern: - {(Finance-Div,AccessPI)} $\subset$ RA - (Admin-Assistant-1,Privacy-Div) $\in$ RU - (Admin-Assistant-1, FinAdminAssistants) $\in$ RU

Figure 7.5 shows the relevant divisions in the enterprise ontology and the *Acts* relation. The enterprise head unit in the diagram is equivalent to the CEO position in the previously presented examples.

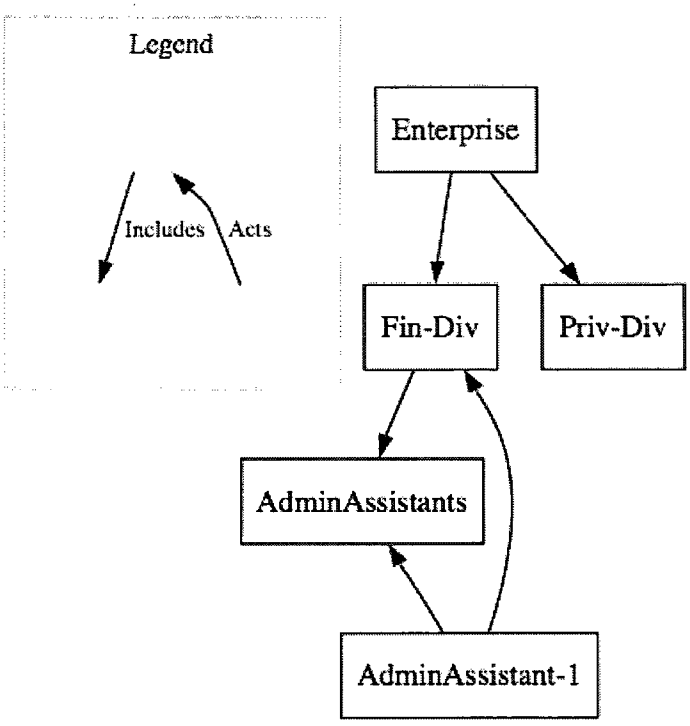


Figure 7.5: Organisational Structure Segment

The finance and privacy divisions are also included in the enterprise (Fin-Div & Priv-Div) and are therefore featured in the diagram, along with the Administrative Assistants role in the Finance division.

Finally, the diagram shows administrative assistant 1 (AdminAssistant-1) acting in the Administrative Assistants role (AdminAssistants) and reporting to the Finance division (Fin-Div).

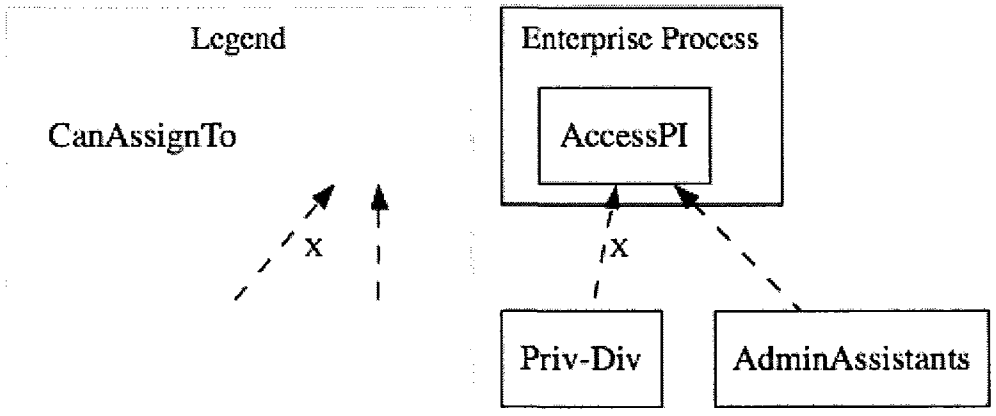


Figure 7.6: CanAssignTo relations

Figure 7.6 shows the *can assign to* relation (CanAssignTo) as specified in the requirements. It further illustrates the privacy process along with the *access personal information* activity (AccessPI). The privacy division (Priv-Div) and the administrative assistants role (AdminAssistants) is also represented. The dashed lines represent the *can assign to* relation.

As per the requirements, it is not possible to assign the Privacy division or role to the *access personal information* activity. The Administrative Assistants role can be given access to the *personal information access* activity

Consistency Validation	
Ontology check	
Logic check	The logic analyser Alloy will validate the above expression to find a contradiction. Since Admin-Assistant-1 belongs to both Finance division and Privacy Division. As per (1) & (2) they both suggest contradictory access.
Scenario check	
Output Analysis	
Violating Model	In this situation, we found a model violation since there is a logical contradiction.
Violating Instance	
Conclusion	A logical contradiction was detected.

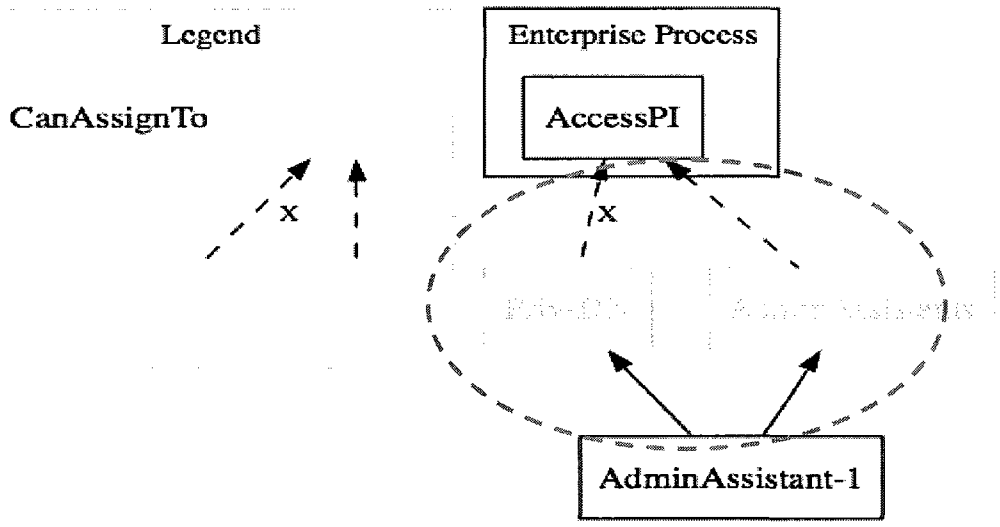


Figure 7.7: Violation Detection

Figure 7.7 shows the results of the analysis. It illustrates how administrative assistant 1 (AdminAssistant-1) is able to gain access through the administrative assistants group (AdminAssistants), but is denied access through the Privacy division relation (Priv-Div).

7.3.3 Example 3: Ontology Sanity

In this example, we will not show a contradiction of values. Rather, we will present the ability of arriving at positively detecting consistency, or in other words, situations where the enterprise is compliant with the law.

Define Requirements	
Legal	1. Privacy process including a privacy audit sub-process 2. An agreement to respect privacy policy needs to be signed by any partner prior to sending information, including personal information 3. All data collection needs to be preceded by a purpose declaration..
Enterprise	4. Audit sub-process as a part of the corporate governance audit 5. All activities in the audit process are assigned to the Financial division. 6. Check the scenario where a subscriber's account needs to be updated with the spouse's personal information.

Figure 7.8 shows the enterprise structure as per the requirements. As in the previous example, the Privacy and Finance divisions (Fin-Div & Priv-Div) are illustrated.

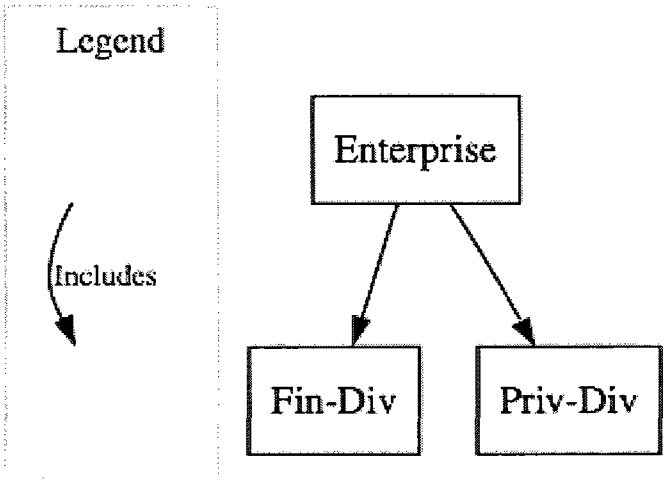


Figure 7.8: Enterprise Structure

Generation

Extract

1. Define a privacy process with an audit sub-process.
2. Define a Finance division
3. Define a corporate governance process with Audit as sub-process
4. Assign the privacy and finance divisions to privacy and financial processes
5. Define activities sign-agreement, sendPI, purpose-declaration, collectPI, updatePI
6. Define activities privacy-audit, finance-audit, SOX-audit as elements of the Audit sub-process

Represent GAL

1) Process Pattern:
   - ComposedOf(Privacy-P, Audit-P)
2) Role Pattern
   - Includes(Enterprise,Fin-Div)
3) Role Pattern
   - ComposedOf(CorporateGov-P,Audit-P)
4) Assignments Pattern:
   - AssignedTo(privacy-audit, Priv-Div)
   - AssignedTo(finance-audit, Finance-Div)
   - AssignedTo(SOX-audit, Finance-Div)
5) Activity Pattern:

- Activity(Sign-Agreement), Activity(Purpose-Declaration), Activity(updatePI)

Represent in Logic $\Phi = \{ P, A, R, U, X,S,D,RA,RU, PA,UP,RR,PP\}$

1) Process Pattern:
   - (Privacy-P,Audit-P) $\in$ PP
2) Role Pattern
   - (Enterprise,Fin-Div)$\in$ RR
3) Role Pattern
   - (CorpGov-P, Audit-P) $\in$ PP
4) Assignments Pattern:
   - {(Finance-Div, privacy-audit),
   - (Finance-Div, finance-audit),
   - (Finance-Div, SOX-audit) } $\subset$ RA
5) Activity Pattern:
   - {Sign-Agreement, Purpose-Declaration, updatePI}$\subset$ A

Figure 7.9 shows the enterprise process in addition to its relation with the organisational structure. The diagram shows that the enterprise process is mainly composed of the privacy (Privacy-P) and corporate governance processes (CorporateGov-P), both containing an audit sub-process (Audit-P). Furthermore, the audit process contains three other processes, namely the SOX audit process (soxaudit), the financial audit process (finAudit) and the privacy audit process (privAudit).

Note that even if the diagram does not show the Audit-Process within the corporate governance process, it nevertheless contains the SOX financial audit and the privacy audit processes.

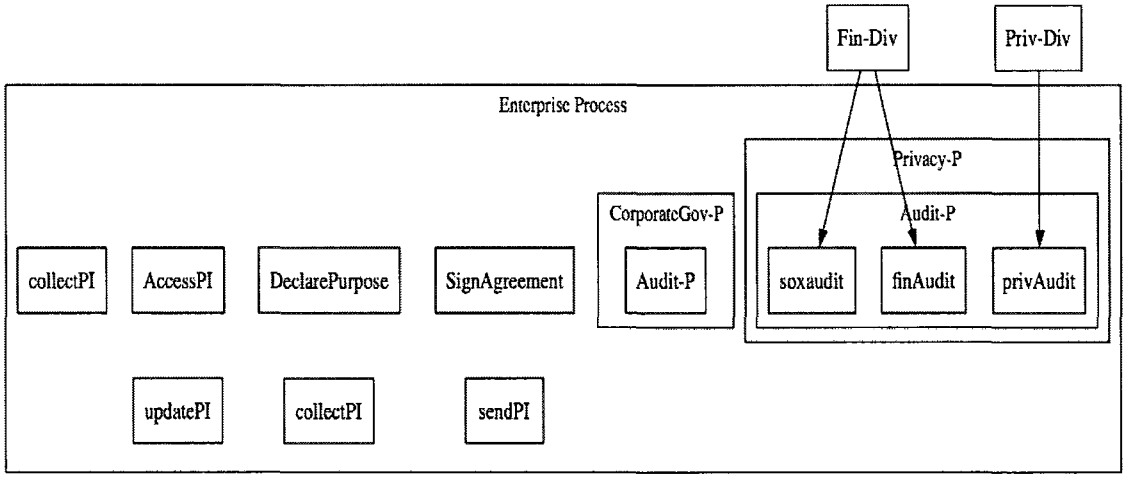


Figure 7.9: Enterprise Process

Finally, the enterprise processes contain the collect personal information, access personal information, declare purpose, sign agreement, update personal information, collect personal information and send personal information activities.

It is also important to mention that the requirements do not explicitly specify further composition of these activities and thus are attributed to the global enterprise process.

Consistency Validation

Ontology check	Assertions:
	Activity-Trace(4.3.1, sign-agreement, sendPI)
	Activity-Trace(4.3.2, purpose-declaration, collectPI)
	The assertions can prove whether the enterprise is consistant with the legal requirements.

Logic check

Con(Φ) In this case the formulae are checked to be true

Scenario check

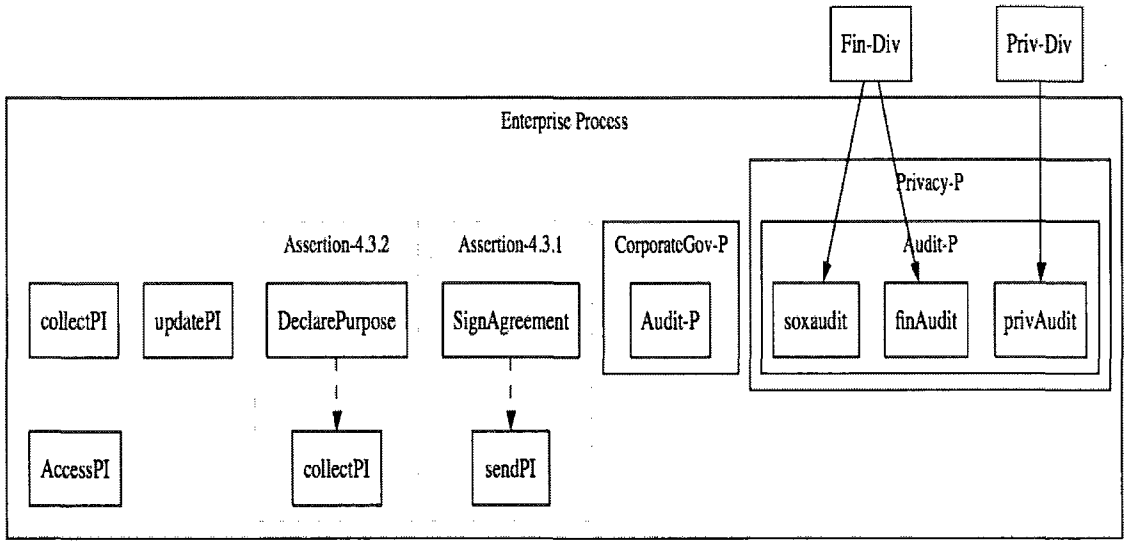


Figure 7.10: Checking assertions

Figure 7.10 illustrates assertions that can be executed by the governance officer to validate requirements. These assertions are illustrated using dotted lines. Assertion label 4.3.1 shows that the requirement to trace a path between sending of personal information and the signing of the agreement. Similarly, the requirements seek to find a path from the collection of personal information to the declaration of purpose.

7.2.4 Example 4: Limiting Use, Disclosure and Retention Violation

Define Requirements: The fifth principle of PIPEDA expressed in provision 4.5 suggests that *personal information shall be retained only as long as the purpose of retention is satisfied.*

Extraction: Process Pattern: Given that the requirement is asking for a precedence relation of activity, we can list the requirement pattern under the process pattern type.

Translate into GAL: The requirement of the GO can be translated into:

Every save-information activity is eventually followed by a dispose-information. This can be represented as follows: Activity-Trace (Provision_4.5, Save_Info, DisposeData)

To signal the accomplishment of the purpose, a thank-client activity may be introduced. Once thank-client is initiated, then data should be disposed of: This can be represented as follows: Activity-Trace (Provision 4.5, Thank Client, Dispose Information).

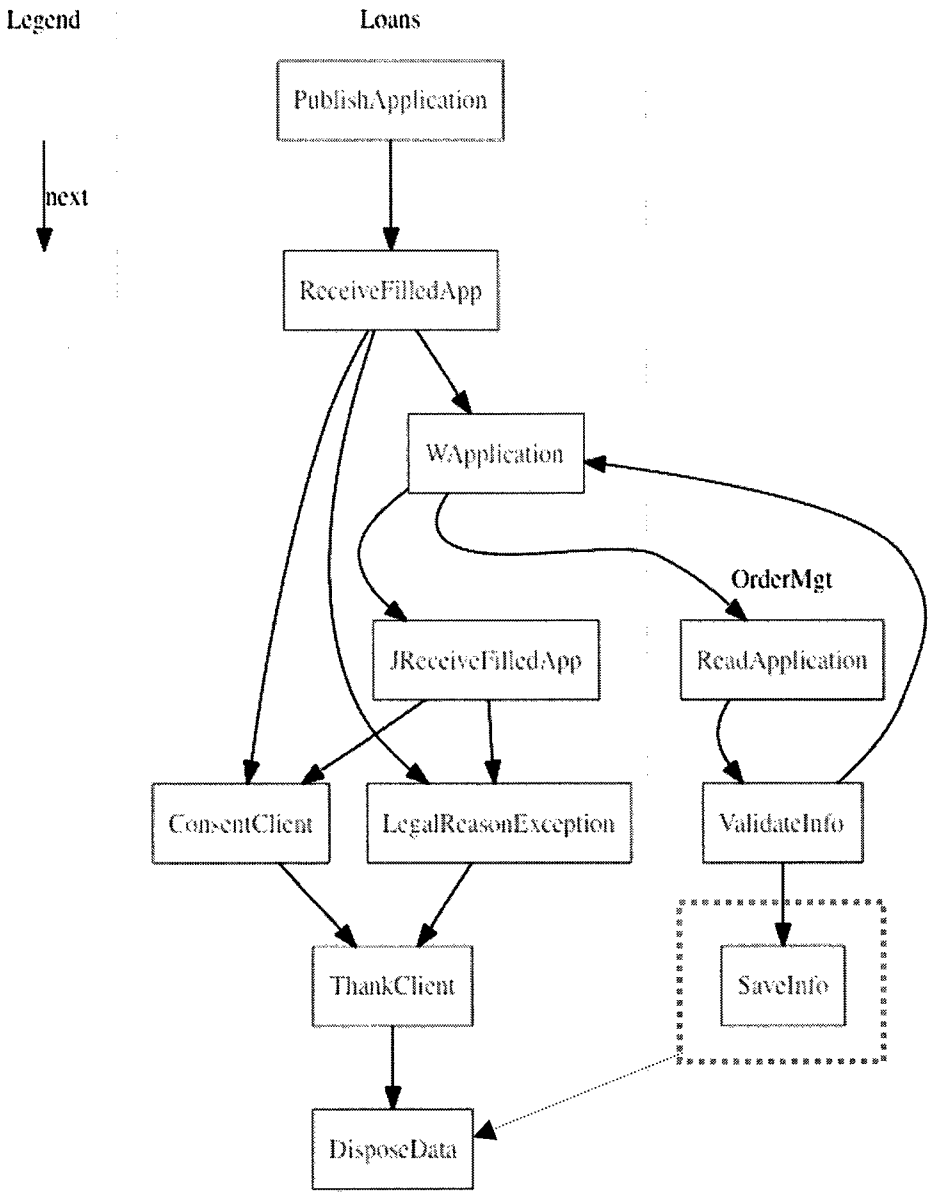


Figure 7.11: Process violation

7.4 Summary and Conclusion

In this chapter, we have shown a proof of concept of the governance analysis method (GAM) and the governance analysis tool (GAT). We have also put forward several examples showing logic, ontology and scenario checks. These examples directly illustrated some PIPEDA requirements and the ability of formal analysers to detect interactions and find assertions in a translated model of the law.

We conclude this chapter by affirming GAM's capacity to check logic, ontology, and scenarios for the purpose of validating privacy compliance.

Chapter 8

Governance Analysis Method for Financial Compliance – Case Study

8.0 Introduction

In the previous chapter, we have shown the ability of GAM and GAT to capture privacy-law requirements and check their consistency within the enterprise implementations. This Chapter shows that the methodology has the ability to address compliance to financial laws.

We will show how financial legal requirements are converted into equivalent GAL statements and present equivalent logic formulae. More importantly, we will also provide a visualized version of our generated model with analyser output. Finally, we will show the results of our analysis and discuss lessons learned. The examples we will use are taken from the American Sarbanes-Oxley Act section 404, Canadian financial law *Instruments 52-109 and 52-111.* However, this selection is not meant to be exclusive in any way, as we are claim that our governance analysis method (GAM) is also able to handle other governance laws with few modifications.

In this chapter, section 8.1 will provide a short overview of financial requirements, as well as a detailed discussion of financial controls, banking transaction life cycle, and separation of concerns, in addition to separation of

concerns issues. Section 8.2 will provide the case study text and section 8.3 will show the implementation of the governance analysis method (GAM). Section 8.4 concludes the Chapter.

8.1 Financial legal requirements

Financial reporting processes are highly driven by IT systems, such systems are used in initiating, authorizing, recording, processing and reporting financial transactions[Fox 2006]. This way, they are linked to the overall financial reporting process and need to be assessed, along with other important processes, for compliance with the Sarbanes-Oxley Act or its Canadian equivalent Instruments 52-109 and 52-111.

The Sarbanes-Oxley Act requires organizations to select and implement a suitable internal control framework. For example, the COSO integrated framework, proposed by the Committee of Sponsoring Organizations of the Treadway Commission is most commonly used for SOX compliance. Since IT control mechanisms have an effect on financial systems, Compliance to financial laws requires enforcing IT control mechanisms to ensure enforcement [Peltier 2005].

In the following subsections we present a definition of financial controls, after which we describe a typical banking transaction life-cycle, we also present separation of concerns in addition to a few examples, finally we extend the discussion by showing some common enterprise policies used to limit risks of SOC rules.

8.1.1 Financial Controls

Financial controls include policies and processes to identify, manage and control operational risk. A key component of risk control are financial policies related to separation of concerns. For example, the Regulation of investment dealers of Canada "100.12, section II.A" stipulates *separation of duties between personnel responsible for submiting a transaction and those responsible for recording the transaction in the books and records.* For an enterprise to be compliant with 100.12, this rule needs to be implemented. Moreover, it needs to be checked for consistency along with other legal and enterprise requirements.

Separation of concerns is a concept that is also applied in the development cycle of software applications [Deek 2005]. The application of internal controls practice demands that functions and system deployment activities such as programming, testing, publishing, operating, controlling and using are performed by different people in order to enhance mutual control [Hugh 2006] .

In the banking sector, separations of Concerns rules are based on the belief that limiting control over multiple types of transaction cycle minimizes risk. In the next sub-section we present details on the banking transactions cycle.

8.1.2 Transaction Cycle

To better explain the banking domain we present the banking transaction cycle[Needles 1993].

Financial transactions are usually divided into several areas of control:

- Authorization—Verifying cash, approving purchases and changes
- Custody—Accessing cash, merchandise, or inventories
- Record keeping—Preparing receipts, maintaining records, and posting payments.
- Reconciliation—Comparing dollar amounts, counts, reports, and payroll summaries

The main claim of implementing SOC controls is the belief in their ability to reduce risk of fraudulent activity by separating:

the custody of assets from accounting personnel,

the authorization of transactions from custody of related assets,

the operational responsibilities from record keeping responsibilities.

8.1.3 Examples of Separation of concerns

Some well-known generic SOC controls[Tohmatsu 2008] are :

- Receiving separated from purchasing and supplier master data
- Requisitioning separated from purchasing
- Purchasing separated from accounts payable and supplier master
- Item master separated from most supply chain activities
- Inventory control separated from accounts payable/settlement
- Separation of purchasing from supplier returns/debit memos

Where the above-mentioned list of controls denies access based on certain rules, there are situations where organisations want to implement resolutions for SOC controls, these are presented in the next section 8.1.4.

8.1.4 Possible resolutions to SOC issues

Whereas separation of concerns rules eliminate risks, they may introduce conflicts [Tarantino 2006] . An enterprise with tens of thousands of users could easily have many conflicts; enterprises with multiple, heterogeneous applications or with multiple organizations operating within their applications. In order to deal with these conflicts, organisations implement what they call resolution policies, which we present in the next sub-section. Some of the business resolutions include several options [Gupta 2008]:

-Forbid the transaction under all circumstances

-Forbid the transaction except with high-level authority

-Permit the transaction based on rules, such as dollar value approval levels

-Permit the transaction with reason codes to justify the action for subsequent review and/or attaching supporting calculations, such as Excel-spreadsheets or application reports

-Permit the transaction with subsequent approval

Removing conflicts can often have an adverse impact on operational performance and introduce delays or errors in a process by involving multiple people [Capdevila 2005]. These rules and others will be subject of future work. In this work we limit the discussion to validating consistency.

8.2 Financial case study

In this subsection, we will present an example scenario of a SOX audit. In section 8.2.1, we present the legal definitions of SOX. In Section 8.2.2 we present a combined mix of SOX and Instruments 52-111 requirements. Finally in 8.2.3, we declare the goals intended for the privacy officer to validate.

8.2.1 Legal Definitions

For the purposes of this case study, we will provide the legal definitions directly from SOX, as they are or with minor modifications.

AUDIT—*The term "audit" means an examination of the financial statements of any issuer by an independent public accounting firm in accordance with the rules of the Board or the Commission.*

COMMISSION—*The term "Commission" means the Securities and Exchange Commission.*

ISSUER—*The term "issuer" means an issuer as defined in section 3 of the Securities Exchange Act of 1934 (15 U.S.C.78c), the securities of which are registered under section 12 of that Act.* The issuer is the firm in question.

NON-AUDIT SERVICES—*The term "non-audit services" means any professional services provided to an issuer (firm) by a registered public accounting firm, other than those provided to an issuer in connection with an audit or a review of the financial statements of an issuer.*

PERSON ASSOCIATED WITH A PUBLIC ACCOUNTING FIRM— (A) IN GENERAL.—*The terms "person associated with a public accounting firm" (or with a "registered public accounting firm") and "associated person of a public accounting firm" (or of a "registered public accounting firm") mean any individual proprietor, partner, shareholder, principal, accountant, or other professional employee of a public accounting firm, or any other independent contractor or entity that, in connection with the preparation or issuance of any audit report .*

PUBLIC ACCOUNTING FIRM—*The term "public accounting firm" means— (A) a proprietorship, partnership, incorporated association, corporation, limited liability company, limited liability partnership, or other legal entity that is engaged in the practice of public accounting or preparing or issuing audit reports; and (B) to the extent so designated by the rules of the Board, any associated person of any entity described in sub-paragraph (A).*

8.2.2 Legal Requirements

We list a limited number of SOX requirements, which set forth the structure and procedures for a given company and affiliated entities.

From Sarbanes-Oxley Act of 2002 (the "Financial Act"): stipulates that the creation of an Audit Committee of the Board of Directors (the Committee) intends to fulfill its responsibilities with respect to the engagement of the independent auditor to perform audit and non-audit services for the Company.

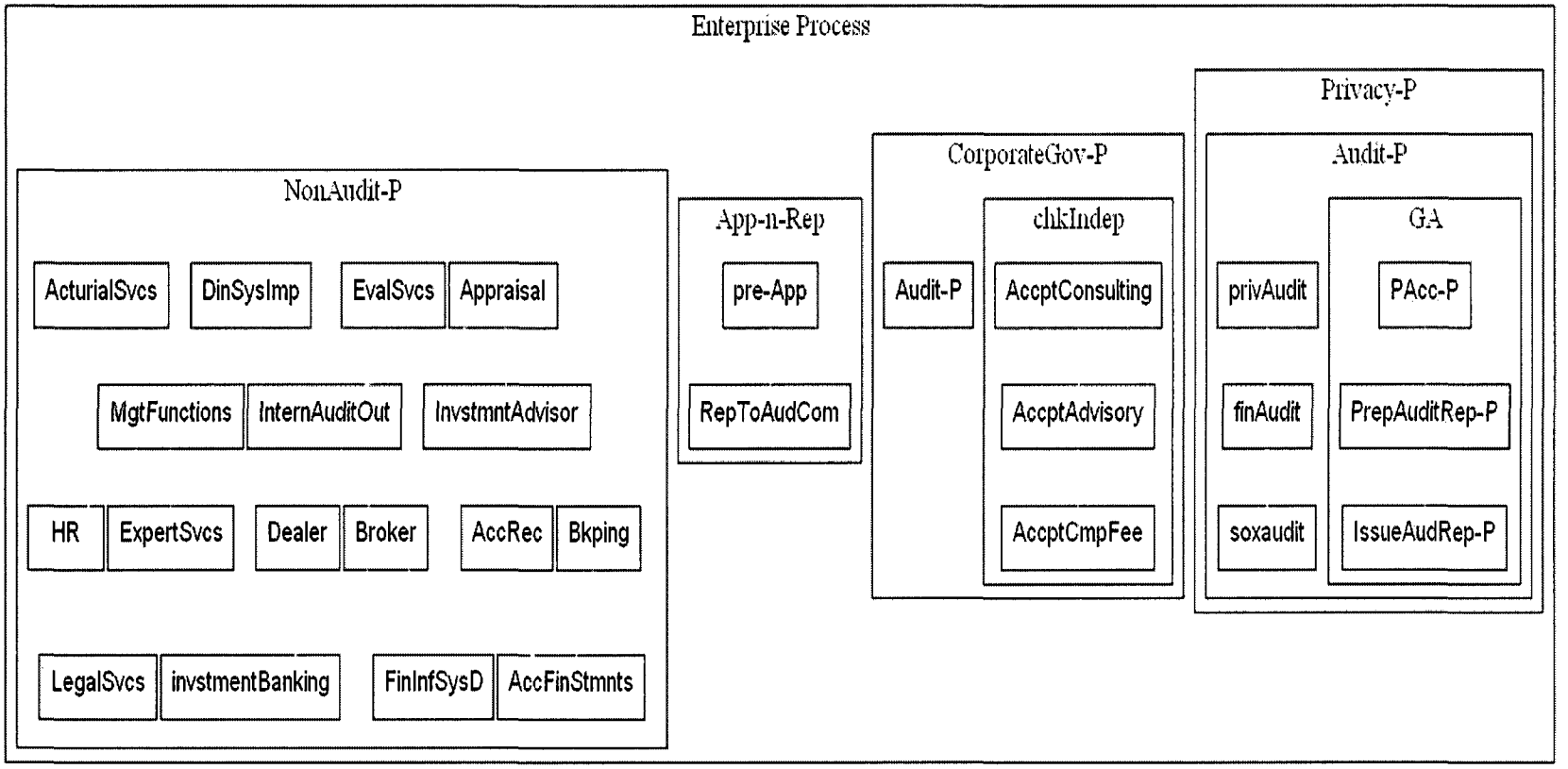


Figure 8.1: Enterprise Process

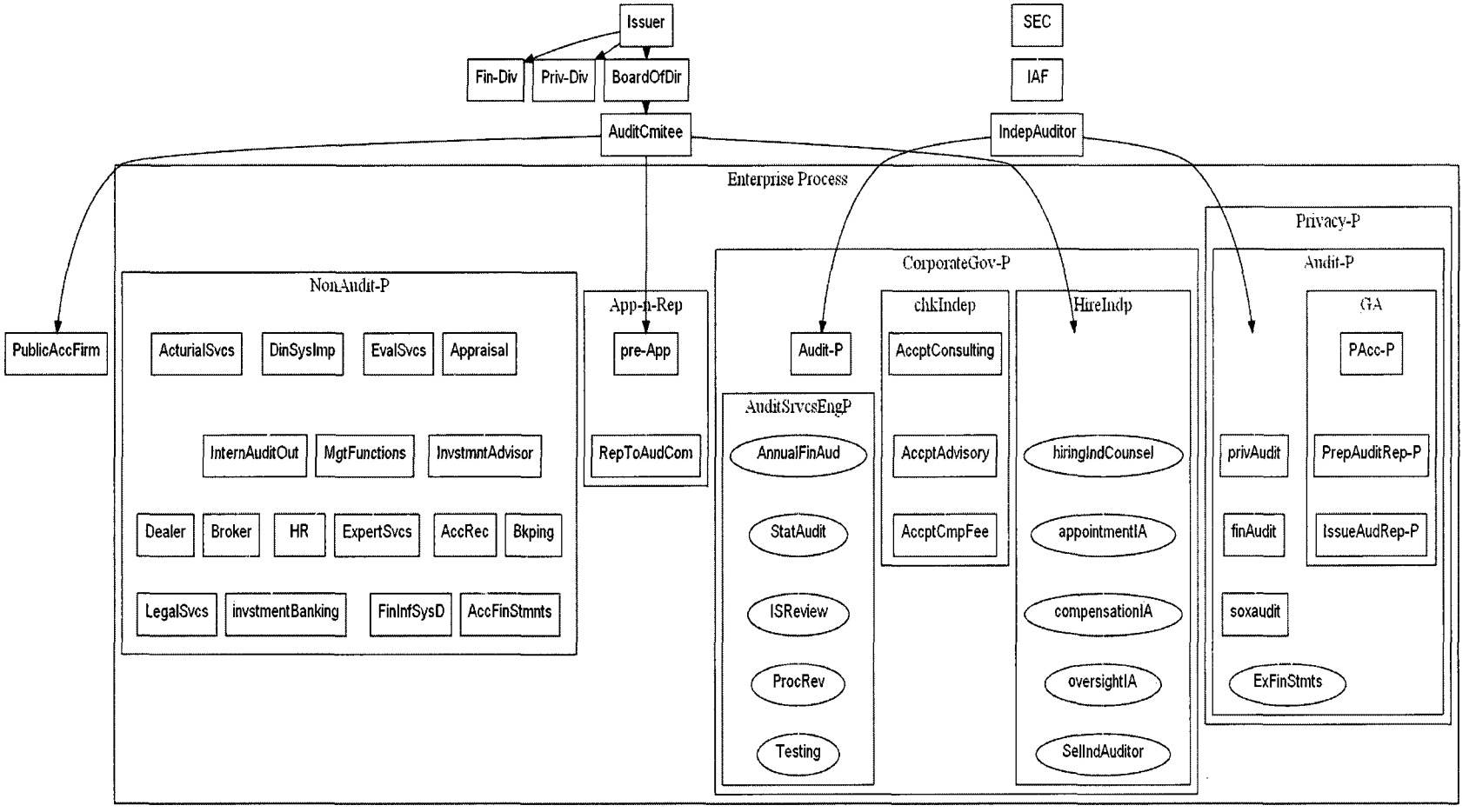


Figure 8.2: Enterprise Ontology and assignments

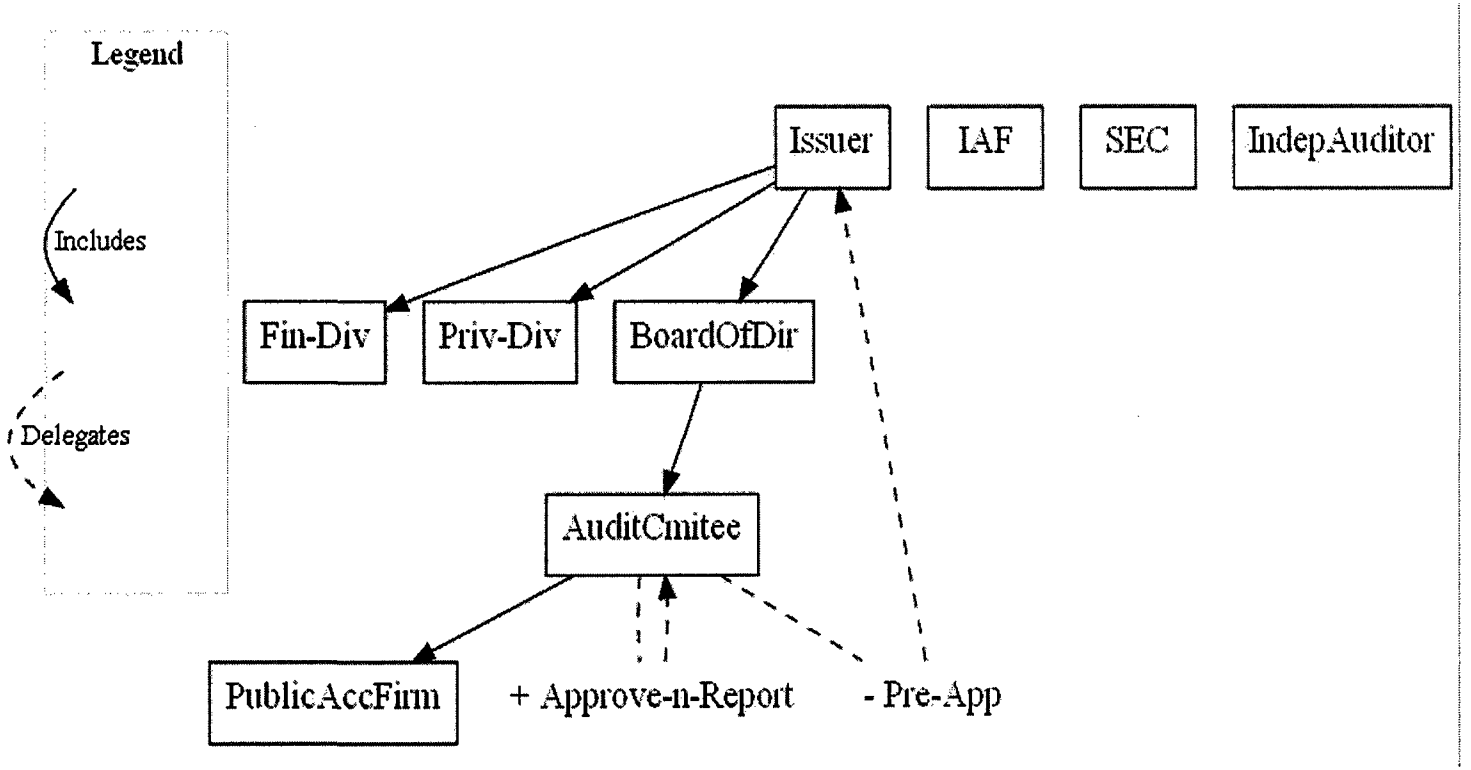


Figure 8.3: Delegation of Authority Rights

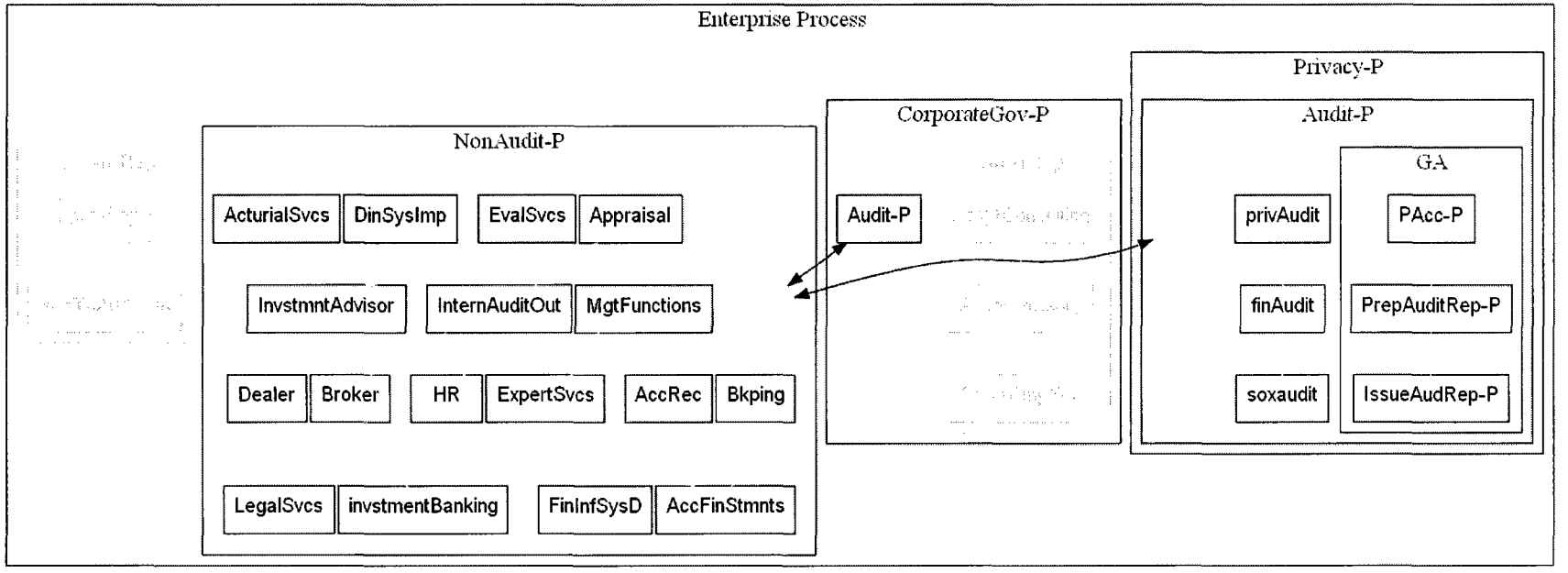


Figure 8.4: Separation of concerns

8.3 Results

This chapter showed a complex enterprise ontology including several departments and business processes, shown in figures 8.1-8.4. In addition, the enterprise model contained separation of concerns rules. This case study has shown that the three techniques of validation proposed in this thesis were shown to be successful in validating legal consistency.

Check 1:

In the first check we, we asked if the public accounting firm reports directly to the audit committee and that the latter is directly responsible for the described function labeled 'Activity'.

The first check was based on ontology. It validated two structural requirements, namely an enterprise hierarchy relation, and a business process assignment relation. The test came back to be successful and hence the enterprise was found to be compliant with respect to the first policy.

Check 2:

The second check is global, it validates if the enterprise requirements derive any inconsistency with legal requirements.

The logic analyser was not able to detect any inconsistencies with legal requirements.

Check 3:

The last check for accessibility of all processes shows the ability of logic analyser to validate all possible scenarios. In this case, we checked if all processes are accessible.

8.4 Conclusion

In this chapter we have shown the ability of the GAM techniques (model, method, and tool) to capture the requirements of financial laws. In fact, our case shows that the formal model can handle many aspects of legal requirements such as separating of concerns and delegation in addition to process and organisational structure specification. We have applied ontology, logic, and scenario checks successfully to detect legal compliance based on logic models. The techniques were shown to be successful in uncovering the results.

Chapter 9

Languages , Tools , Evaluation

9.0 Introduction

This chapter presents performance analysis tests of the analyser models produced by GAT. We use several metrics to measure performance of model consistency and scenario checks suggested in our method.

In the first section, we will present an overview of the Alloy language used throughout the thesis. Section 9.2 will present the satisfiability problem and the SAT solvers used in the benchmarks. In section 9.3, we will present our testing methodology and in section 9.4, we will present a summary of results. In section 9.5, we present test implementation and detailed observations.

9.1 Alloy

In this section we will provide an overview of Alloy, in addition to small set of language constructs, and finally some discussion of characteristics and limitations.

9.1.1 Overview

Formal methods stem from mathematics and aim to help design, develop, analyse and validate software in order to make it correct, error-free and robust [Bowen 2008]. Formal models are built on basic elements such as sets and set operators, and can be analysed against precisely defined properties like

consistency. Formal methods include Petri nets, abstract state machines, process calculi, temporal and belief logics, as well as languages such as Z [Z 2002], CSP[Roscoe 1997] and Alloy[Jackson 2003]. Formal methods offer solid methods, produce clear models and have improving tool support.

Using logic to model legislation has been one of the early techniques in representing legal requirements. Such an approach compensates for the irregularities of natural language by using concrete concepts. Previous techniques employed logical connectives (implication, conjunction, and so on). Employing logic to fill the gaps was criticized as limited. In this thesis, we have taken the approach of using first order logic combined with ontologies to capture and validate legal compliance. Our quest for a formal tool was satisfied by Alloy-4.

Alloy is one of several emerging "lightweight formal methods" that uses simple first-order logic for expressing designs and properties coupled with constraint-solving technology for automatic analysis.

Alloy-4 consists of a formal language and related model analyser and therefore defines a formal method that can be used to precisely capture and analyse logical specifications of systems.

The Alloy language is a simple structural modeling language based on first-order logic. The model analyser can generate instances of invariants, simulate the execution of operations and check the user-specified properties of a model.

There are three basic levels of abstraction in Alloy modeling. At the highest level, Alloy follows an object-oriented paradigm. The middle level is the set theory level where a model is represented in terms of sets and relationships. At the lowest level are atoms and relations, which correspond to the true semantics of the language.

The Alloy-4 tool is a logic analyser and model finder accepting specifications written in Alloy as input. The tool can generate instances of invariants, simulate the execution of operations and check the user-specified properties of a model. Alloy-4 is modular and extensible. It has a core relational logic engine that incorporates new optimization techniques. The logic engine can be accessed in two ways. Either a compiler allows the model to be expressed in textual form or a set of Java™ API methods allows the model to be constructed, queried and analysed dynamically. If it is beneficial, additional interfaces can be easily written to integrate it into another analysis framework.

The Alloy Analyser 4 uses Kodkod [Narain 2008], an efficient SAT-based analysis engine for first-order logic with relations, transitive closure and partial instances. Optimizations are first performed at the Alloy level with the reduced problem then being given to Kodkod.

Alloy is a first-order, declarative language suitable for expressing models of software systems. Alloy models are open to fully automatic analysis, using the Alloy Analyzer. The analyzer translates Alloy formulas to propositional formulas using a given scope, i.e., a bound on the universe of discourse, and uses

off-the-shelf SAT solvers to find concrete instances or counterexamples for Alloy formulas. Alloy has been successfully used in research and teaching for several years and has assisted in finding and correcting design flaws in various systems.

An Alloy model is a set of classes defining the basic relationships. Classes are bound by relationships. Functions and facts set predicates. Finally, assertions are run to verify the validity of a ruleset.

9.1.2 Language constructs

1. Signatures: A signature is like a class in UML, or record in Pascal. It is a basic unit signifying a relationship with its members.

2. Relationships: Everything in Alloy is a relationship. Signatures are relationships and can have relationships between themselves. Alloy offers the possibility of discovering reverse relationships using the operator (~). Example: if F: A-B, then ~B is the set of all elements that have a relationship with B in A.

3. Facts: Facts are uncontested invariants. A system with contradicting facts cannot be instantiated. Facts are asserted as invariants using the keyword fact. Facts can constrain values or relationships.

4. Assertions: Assertions are used as questions to find out if a rule is violated. This can help detect known interactions. For example, a question can be if it is possible that employees of the credit department have access to the loans department.

5. Operators: Alloy supports the regular set-theoretical operator set. [+,&,-] set union, intersection, and difference, in addition to the set quantifiers

[all, some, none, sole, one]. Most important, we address the dot composition operator ["."]. The dot operator has multiple significations depending on the level of abstraction used. Composition ["."]: The composition operator ["."] works as a join between two tuple sets. For example, Set S1(A,B) can be composed with set S2(B,C) to produce S3(A,C). It can also be used to dereference a relationship in the class.

6. Multiplicities: Any relationship can have a multiplicity in Alloy. Multiplicities are crucial because they tell the Alloy analyser how many instances to create. For example, there can be 3 roles, 15 employees and 20 contractors. This instructs Alloy to create such a model with the required constraints. Scope: Knowing that computing the consistency of the model depends on the arity and the multiplicity of objects, Alloy allows to specify the size of an instance space during execution. This allows users to say: run my world of 5 object classes and 50 instances with their corresponding relationships, to see if the model is consistent.

9.1.3 Alloy characteristics and Issues

The rule-based nature and object orientation of Alloy are suitable for the enterprise model under consideration. A very important property of Alloy is the fact that it allows to limit the instance size. On the other hand, we emphasize the fact that our process based methodology and our claims are tool independent, and can be investigated using other analysers.

Explosion avoidance

A special feature of Alloy is that it can populate a world of possibilities that is constrained by a specific size. One can then model a system with a fixed number of entities. A system designer can leave it up to Alloy to populate

instances randomly and non-deterministically. However, an inconsistency that does not show in a certain model size could very well appear in the next one, hence it is important to try as large a size as possible.

Model consistency

Given a model, if it is inconsistent, Alloy can detect the inconsistency but it does not specify the reasons for inconsistency, nor does it provide a counterexample. Therefore, when building a model, one needs to incrementally verify the system, i.e. the test for consistency must be repeated every time the model is changed, otherwise the reason for the inconsistency can become difficult to detect. It is left to manual labor to decide the cause of the inconsistency.

9.2 Satisfiability problem and SAT solvers

Alloy offers an easy to use front end to SAT solvers, which turn the problem into a satisfiability problem.

Satisfiability is the problem of determining if the variables of a given Boolean formula can be assigned in a way to make the formula evaluate to TRUE. It is equally important to determine whether no such assignments exist, which is equivalent to say that the function expressed by the formula is identically FALSE for all possible variable assignments. In the latter case, we say that the formula is *un-satisfiable*. With relation to terminology used elsewhere in the thesis, an unsatisfiable formula is *contradictory* or *inconsistent*, or an unsatisfiable formula *has no model*.

This problem is frequently called: Boolean or propositional satisfiability. The shorthand "SAT" is also commonly used, with the implicit understanding that the function and its variables are all binary-valued. The Boolean satisfaction algorithm is known to be NP-complete, and the best algorithms that are known for it are exponential. However there is much research on efficient algorithms for the problem, and there are tools to solve it for a limited number of variables, are we will see in this chapter.

In order to evaluate any potential speed and performance differences between the various solvers, we studied three solvers ZChaff, MiniSAT and SAT4J . These solvers are already bundled with the Alloy tool.

ZChaff is an algorithm for solving instances of the Boolean satisfiability problem in programming. Designed by researchers at Princeton University, ZChaff is a popular solver whose source is available to the public. It implements the well-known Chaff algorithm [Moskewicz 2005] which includes the innovative VSIDS decision strategy and a very efficient Boolean constraint propagation procedure [Madigan 2001].

MiniSAT [Sörensson 2006] is a small, complete and efficient SAT-solver in the style of conflict driver learning, as exemplified by Chaff. It provides the ability to make specific extensions or an adaptation of the techniques provided and also includes a mechanism for adding Boolean constraints.

The aim of the SAT4J library is to provide an efficient library of SAT solvers in Java. This library has been used in problems of various complexity including memory leak detection, relational databases, and requirements driven software systems.

9.2 Testing the Governance Analysis Methodology

In this section, we will describe our goals and method in testing the governance analysis method. In section 9.2.1 we will set our goals, in section 9.2.2 we will present our control parameters, and finally in section 9.2.3 we will list qualification measures.

We have used as a basis for our tests the case studies of Chapters 7 and 8, and then we have modified their complexity to obtain the results that we will show.

9.2.1 Goals

A major goal of testing the methodology is to measure the basic ability of a method such as GAM to address consistency problems. In addition to testing the ability of the models, we intend to test the performance of the logic analyser and its embedded SAT solver to check consistency and scenario violations.

We intend to study the following questions:

Table 9.1: Testing Goals and Descriptions

Goal Name	Description
Consistency check time as a function of the number of variables	To study the effect of increasing the number of variables on the time taken to check consistency for a given model
Scenario check time as a function of the number of variables	To study the effect of varying the number of variables on the time taken to check a scenario for validity or to provide a counterexample
Consistency check time as a function of complexity	To study the effect of model complexity on the time taken to check for consistency
Scenario check time as a function of complexity (lower bounds)	For small to medium size models, we have studied the effect of complexity on the time taken to check a scenario to provide a counterexample. These tests were executed on models featuring a small number of users.
Scenario check time as a function of complexity (upper bounds)	Similar to the test above. However, in this test, we wanted to observe the effect of varying complexity for larger models
Scenario & Consistency	We wanted to show the effect of complexity for

check time as a function of complexity (upper Bounds	large models on scenario and consistency checks plotted together.

9.2.2 Metrics

After each test we have analysed the following qualifiers:

- Time taken to validate the consistency of the model.
- Time taken to validate a scenario or an ontology check.
- Termination, or whether the analyser was able to successfully terminate with a result and within bounded scope.
- Stress, the ability to handle a combination of a large number of variables or complex formulas.

Table 9.2 describes the test coverage presented in this chapter. From left to right we list the qualifiers: these are the consistency time, the scenario time, termination, and stress test.

The scenario time is the time taken by the solver to check the validity of a given scenario.

Consistency time is the time taken by the solver to check the consistency of a given model.

Termination is the ability to arrive at a conclusion for either the scenario and logic check. If the solver crashes or hangs, this is considered to be a non-termination. However, if the solver exits declaring that solver could not validate a scenario then this situation is considered as a termination.

Table 9.2: Testing Metrics

Behaviours / Variables	Consistency Time	Scenario Time	Termination	Stress
Number of Variables	X	X	X	X
Formula Complexity	X	X	X	X
Variables & Complexity		X	X	X

The stress test checks whether the abilities of the solver were tested using large number of variables, classes, and formulas in addition to increased complexity of the formulas.

Throughout the tests we manipulate some factors that are: Number of variables, Complexity of formulas, and a combination of number of variables and complexity.

Varying the number of variables provides us with the ability to observe the behaviour of various solvers as the number of variables in the logic model is increased.

Varying the complexity of the formulas provides us with the ability to observe the behaviour as the model complexity increases.

Varying both, the number of instance variables and the complexity, helps understand the behavior of the logic model as both factors (size and complexity) are increased.

The behaviour of the SAT solver was studied as we checked consistency, scenario, termination, and stress factors.

9.2.3 Parameters

We had control of several parameters for the purpose of evaluating the performance of the output models.

Throughout the tests we changed the following parameters.:

- Allocated Memory: Through the tool, we were able to control the amount of memory reserved for the logic analyser
- Number of Classes: We executed tests with a varying number of requirement classes.
- Number of Instances (specified and generated): In many of the tests, we changed the number of instance variables.
- Number of Formulas: We tested the model with a varying number of

logic assertions.

- Type of Formulas (Joins, Closure, Union, Subtraction): We tested the model by changing the complexity of the formulas
- Choice of SAT solver (SAT4j , ZChaff, MiniSAT): All of the above variables were tested against the three SAT solvers.

The summary below shows only some essential findings.

9.3 Summary of lessons learned

1. SAT solvers differ in their ability to handle logic models. However, these differences among the test solvers were not large enough to substantiate a performance edge. Although there were obvious limitations for certain solvers causing the analyser to crash, we were able to derive a conclusion each time the solvers ran successfully. In other words, the analyser was able to either satisfy the problem or validate the asserted formula.

2. Models containing multiple complex formulas containing logic closures and joins caused major delays and higher uses of memory. Combining negative and positive assertions in formulas expanded these delays.

3. The amount of memory needed was also highly influenced by the number of classes of objects.

4. Depending on the complexity of the model, there is a limit on the number of variables the model can actually instantiate. This limit is imposed by the ability of the solver to produce a result before running out of memory.

5. Having more instances meant higher uses of memory and delays in the consistency checks.

6. For a large number of instances, the increase of logic rule complexity did not cause a significant delay.

9.4 Observations

In this section, we will provide details of the setup for the test, observations and a summary of results. Times given are in milliseconds as reported by the analyser. Note that actual times for showing a graphical result may require a few more seconds.

9.4.1 Setup

The tests where executed on an iMac computer with an operating system of MAC-OSX-10.5.5 Leopard version, a dual core Intel processor, 4 MB of Layer 2 cache and 3GB of RAM running on 800 MHZ bus speed. The system had several other desktop applications and an operating browser during the tests.

9.4.2 Results

Consistency-check time as a function of variables

In the first test, we measured the influence of the number of variables defined in a model on the amount of time taken to come up with a consistency solution.

In Figure 9.1, we have mapped time as a function of the number of variables. The number of variables is plotted on the X-axis and the time in milliseconds is plotted on the Y-axis. The colored lines represent the three solvers we have used, namely MiniSAT, SAT4j, and ZChaff. MiniSAT and ZChaff appear to have lower and rather similar results than SAT4j

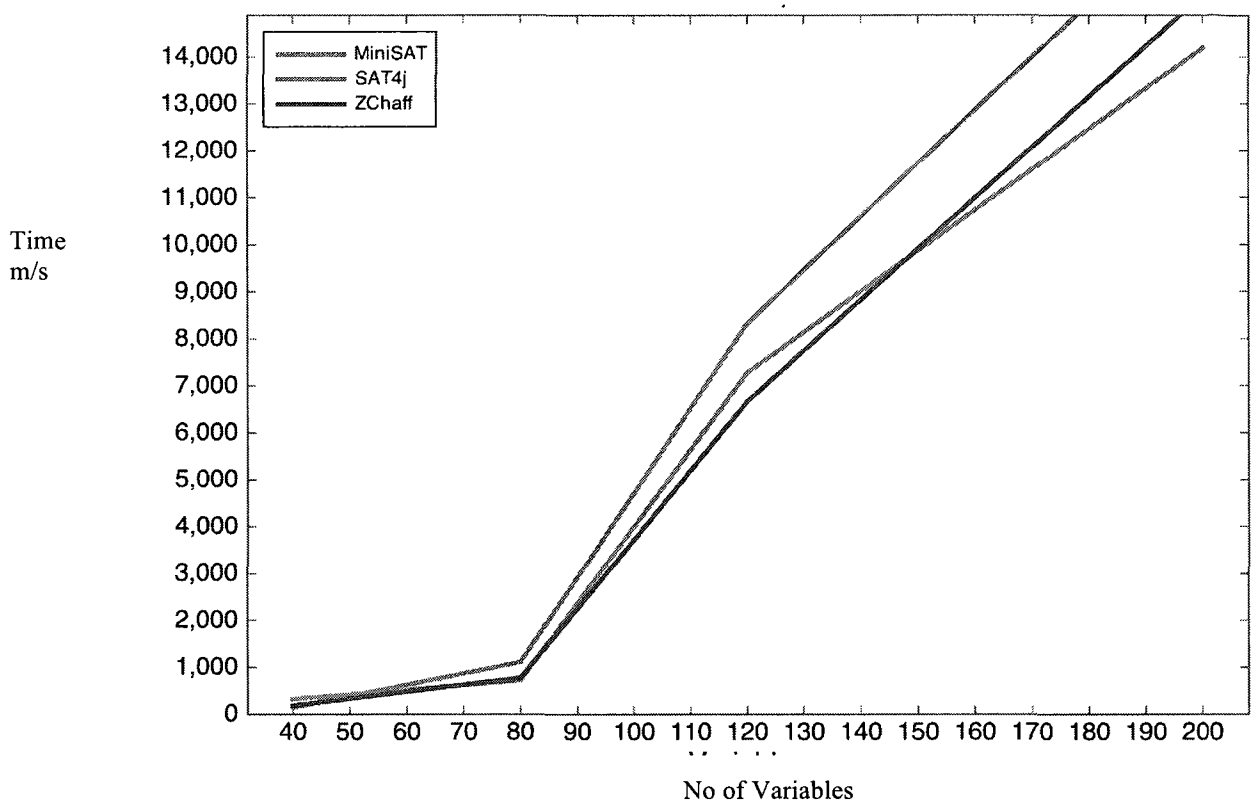


Figure 9.1: Time for testing consistency as a function of the number of variables

Scenario-check time as a function of variables

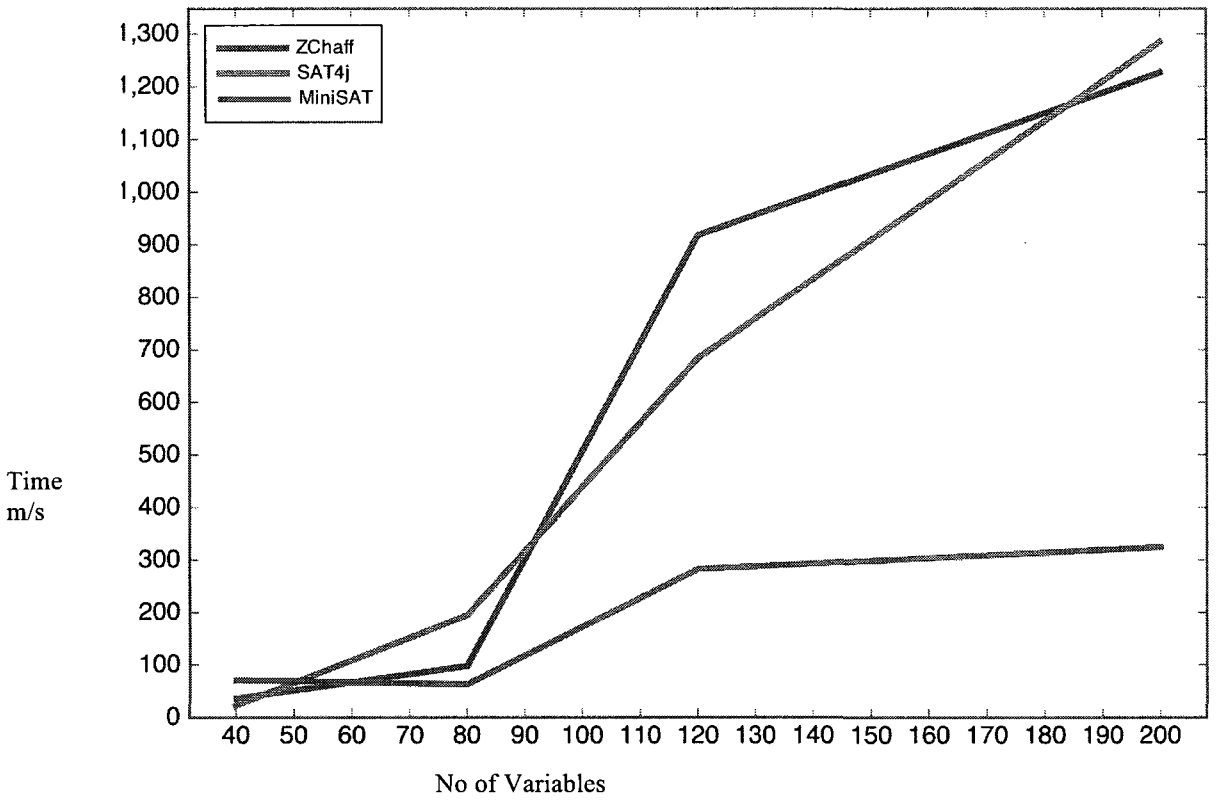


Figure 9.2: Measuring Time delay of Scenario check as a function of variables

In Figure 9.2, we plotted the time taken to solve a scenario as a function of the number of variables. It appears that for the scenario checks, MiniSAT performed better than both SAT4j and ZChaff, which are in tandem for most of the graph. The graph also shows that all three solvers have a similar delay for smaller numbers of variables.

Consistency-check time as a function of complexity

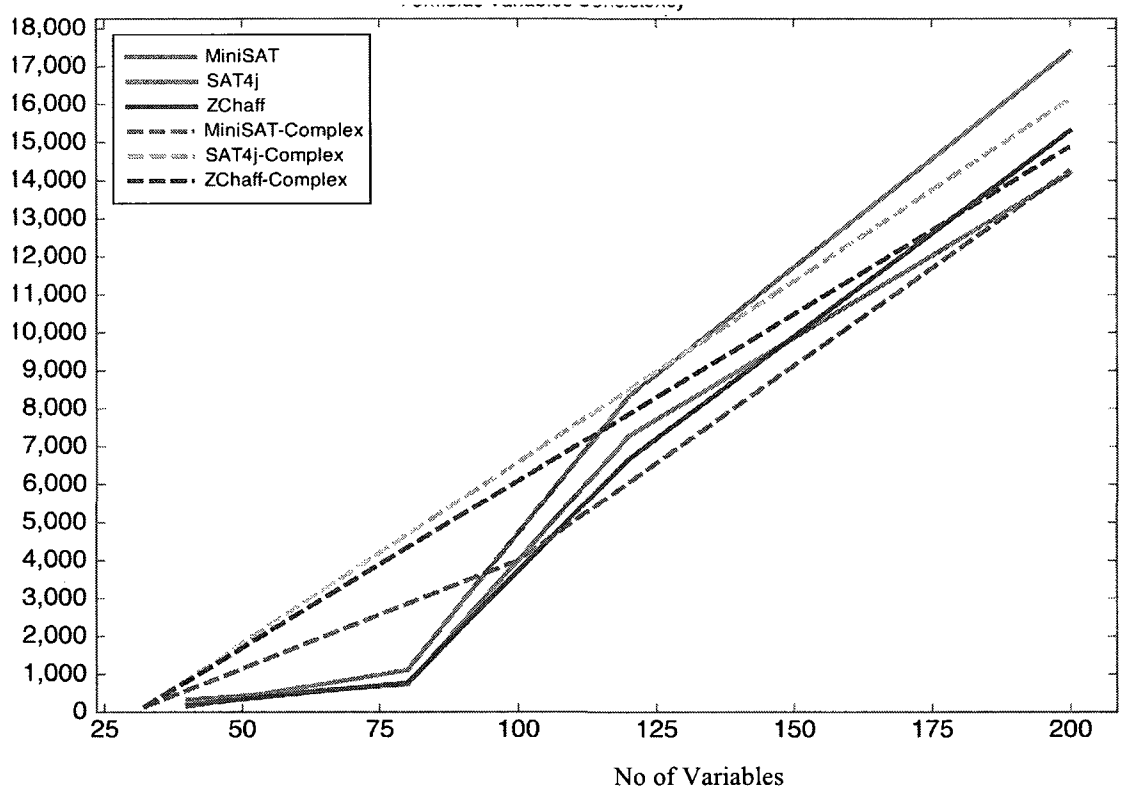


Figure 9.3: Measuring Time Delay of complex formulas in consistency check

In Figure 9.3, we illustrated the time delay when measuring consistency based on varying both the number and complexity of formulas.

The set of solid lines represents the time taken to solve the consistency of the model whereas the set of dashed lines represents the time needed to solve the consistency of the model after adding complex formulas.

As it clearly appears in the figure, the complexity of the formulas did not affect the time needed to solve the issue of consistency.

Scenario-check time as a function of complexity (lower bounds)

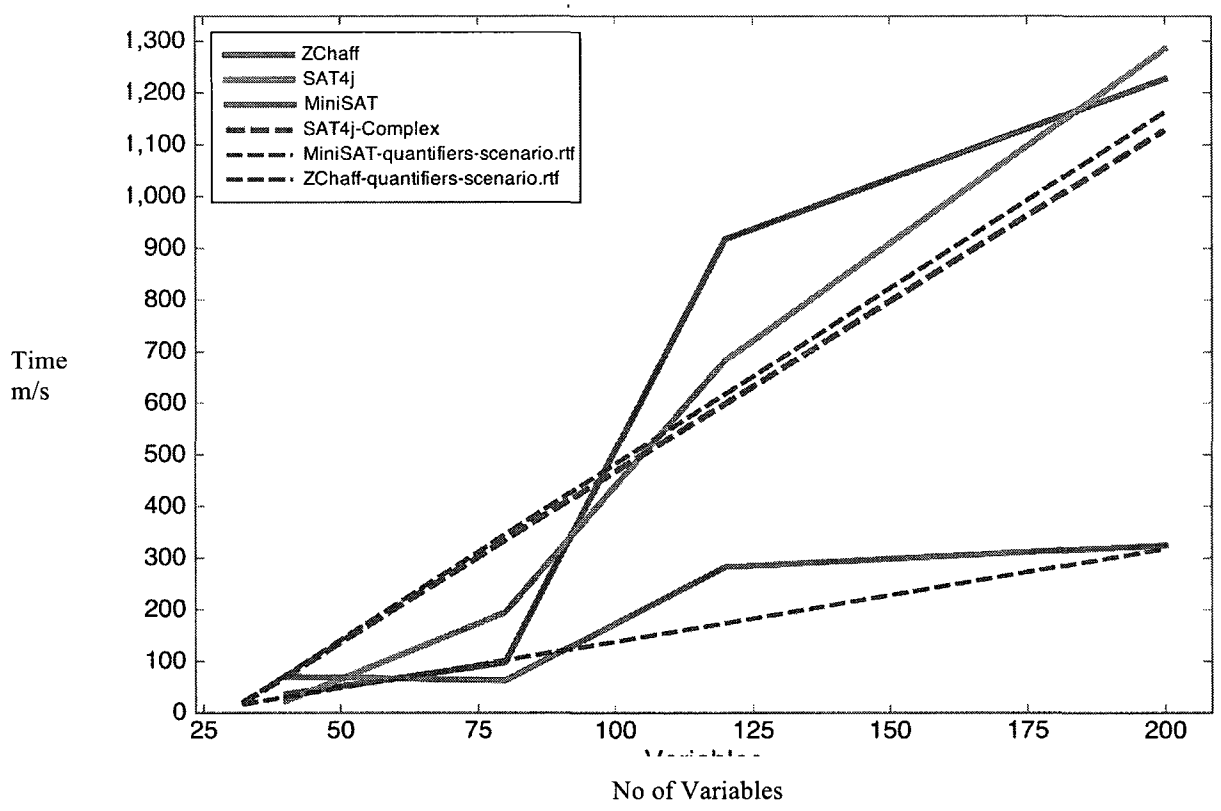


Figure 9.4: Measuring Time delay on complex formulas in scenario check

Figure 9.4 shows the relative relationship of the number of variables to the time consumed to check scenarios of varying complexities on smaller bounds, *i.e.* variable sizes of 200 or less. As in Figure 9.3, the complexity of the formulas did not affect the behavior of tools with respect to finding a solution for the scenarios.

Scenario-check time as a function of complexity (upper bounds)

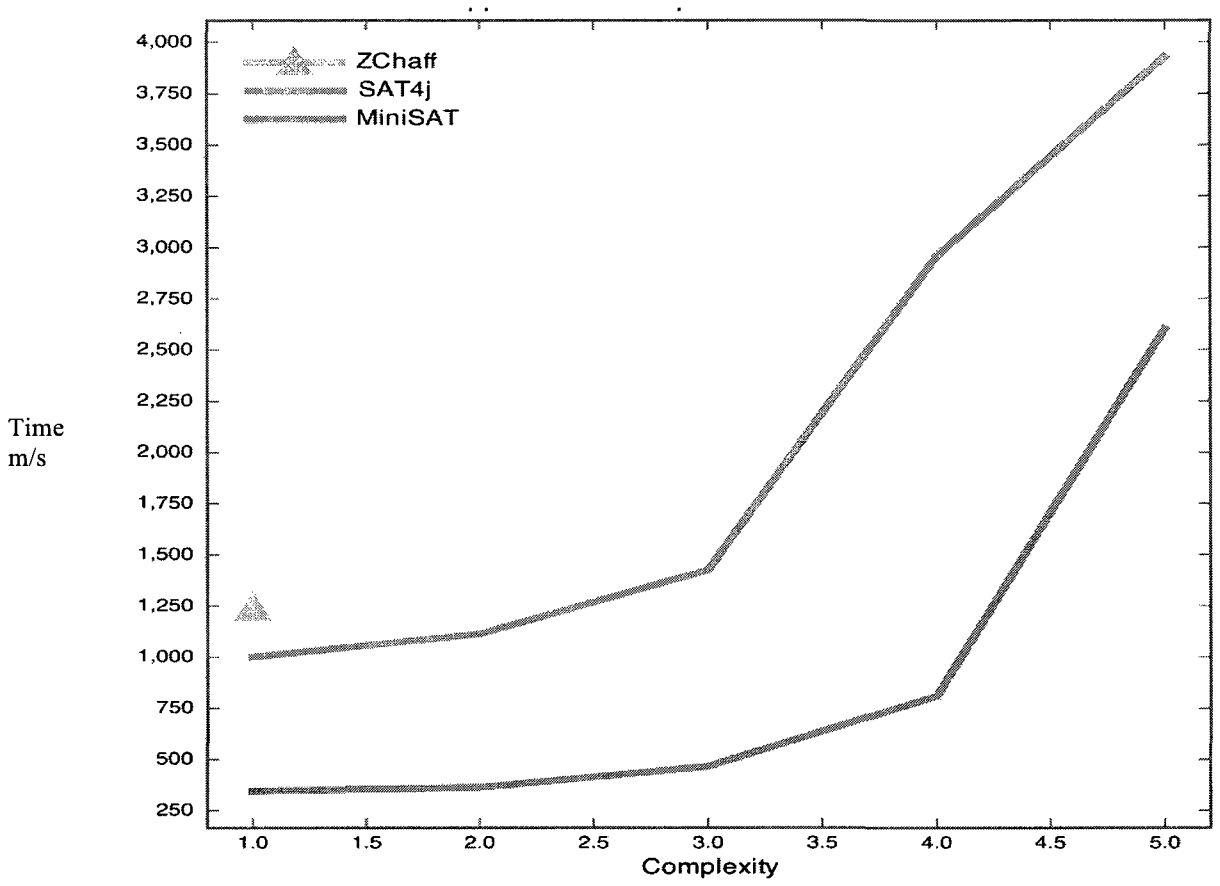


Figure 9.5: Measuring time delays of complex formulas in scenario check

On upper bounds, the number of variables being around 400.

In this test, we fixed the number of variables while increasing the complexity of the scenario formulas. As you can see in Figure 9.5, we represented the complexity on the X axis and the time on the Y axis.

ZChaff crashed and did not terminate in this group of tests, whereas SAT4j's performance was lower than MiniSAT overall.

Scenario & Consistency-check time as a function of complexity (Upper Bounds)

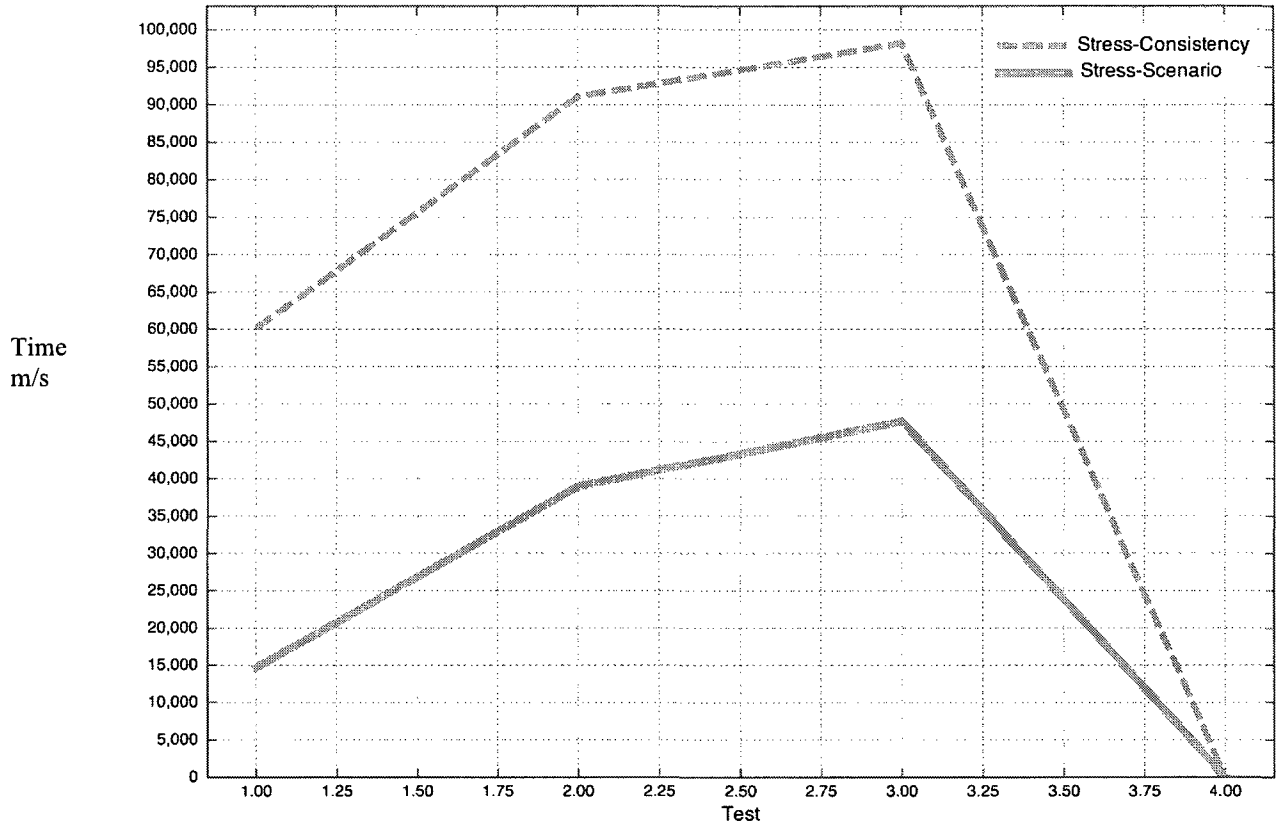


Figure 9.6: Stress testing on consistency and scenarios

Figure 9.6 shows the effect of stress testing on SAT4j. The X-axis represents various tests with the number of variables and classes increasing as we move along the axis towards to the right. The dotted line in the graph shows that the consistency test always needs more time than the scenario test. However, both crashed after test number 3, this is represented in the declining line.

9.5 Addressing Challenges

In chapter 1 we listed several challenges facing privacy and formal logic research. In this section we will discuss the extent to which our work has addressed the listed challenges.

9.5.1 Method for requirement definition

We previously referenced the challenges of requirement definition. Laws are written in natural language and are open to interpretation owing to interdependencies and a possible lack of precision. Our extraction method assists a specialist in extracting requirements by providing 10 patterns. The activity, legal entity, process, role, assignment, delegation, and separation of concerns patterns are explicit. Hence can potentially facilitate the extraction process. The higher-level patterns we propose are composite, declarative, and definition. Higher-level patterns are not analysed in detail, a study and classification of such requirements is needed.

We conjecture that the advantages of proposed extraction method can only be realized after a graphical user interface and a user study is conducted to evaluate effectiveness.

9.5.2 Method for checking consistency

Our method proposes three techniques to validate consistency of legal and enterprise requirements: Logic, ontology, and scenario checks. We proposed basic definition of consistency. Based on these definitions we defined the three

techniques formally and provided examples. We have shown that the techniques are reasonably capable in detecting inconsistencies based on defined requirements.

9.5.3 Technique for defining ontology

Laws as introduced earlier are rich in ontology definitions. Process and enterprise ontology are used in financial and privacy laws. GAM supports both definitions throughout the reference model, the method, language, and checks. In the reference model there are separate planes for enterprise structure and process. The GAL provides elements to define ontology elements and their relations. Further, the consistency validation method includes checks for ontology compliance. By way of examples, we show that the process of ontology definition and validation is possible.

9.5.4 Method for checking completeness

The difficulty of checking completeness lies in its definition and implementation. Completeness in the legal sense, as proposed in chapter 3, means that the enterprise is compliant with the scenarios defined in the law. In addition, the enterprise specification should be able to cover all possible cases possible under the law. Our work has provided a definition of completeness in computer terms. Furthermore the scenario checking technique was able to detect conflicting scenarios. Note that our method to validate completeness did

not include the coverage aspect. Hence our method needs further analysis and implementation of coverage checks.

9.6 Other research metrics

9.6.1 Dealing with change

In section 4.5.2 we have proposed a change management process. There are two types of changes, incremental, and relational changes. All changes require a reinterpretation of the GAL statements. However only relationship changes require consistency checks. Validation and interpretation processes are automated, hence this facilitates change management.

9.6.2 Resolving differences of terminology

Often enterprises have their particular terminology, which may be different from the law. Such terms are critical to the business process, enterprise structure and related activities. Following GAM, the terminology issues are resolved by way of ontology definition. One of the offered predicates through GAL links related terms. Following the linkage, the formal analyser is able to recognize the various terms and reason accordingly.

9.6.3 Skills needed to apply method

Users of GAM should combine three types of expertise following the GAM method. Reading the law and using the patterns to extract requirements, writing the requirements in GAL instructions, which requires knowledge of

GAL, and finally validating requirements using the Alloy tool. It is certain that the expertise is particular, however, we conjecture that the learning curve is not steep. Future work could take the method and validate its effectiveness through an empirical case study.

9.7 Related work

From the related we identify three groups of related work of Ghanavati et al., Antón et al., Onabajo et al. We will look at related work based on two dimensions: goal, and direction.

9.7.1 Goal –Validating Compliance

The work of Gahnavati et al. [Ghanavati 2009] [Ghanavati 2008] [Peyton 2007] [Ghanavati 2007] [Weiss 2005] converges towards the validation of compliance of enterprise requirements to legal requirements. Their work is still in early stages, however they have shown results on two aspects, the ability to provide traceability from legal requirements to enterprise requirements. Furthermore, they are able to validate compliance of static requirements. Our work provides the advantage of formal analysis of dynamic requirements. Furthermore, our use of first order logic provides semantics to our process models. Using logic model we are able to group requirements and reason in terms of their groupings. Another advantage of our work is that we have defined the domain including terms of compliance, consistency, and others.

9.7.2 Approach – requirement extraction and interpretation

The work of Onabajo et al. has the same direction as ours. They provide methods to extract requirements from natural language and define their semantics in computable models.

[Onabajo 2009] presents the CREE method supporting:

1) Goal Annotation: Goals and their relationships are identified in the natural language documents and annotated with concepts from a meta model. Our method, on the other hand does not include annotations.

2) Structural Analysis: Structures of the annotated goals are analyzed with respect to their structural completeness. This covers two cases: omission by analyst, ambiguity in source document. This aspect is covered by our ontology check technique.

3) Terminological Sorting: Annotated concepts are consolidated into consistent terminologies. This aspect is supported by the -GAL- ontology statements. These statements define relations between ontology elements.

4) Semantic Analysis: Goal models are analyzed for semantic consistency. In their work, two cases are considered: Inferring requirement goals, Interference, Conflict and Exception identification. They use defeasible logic for internal representation of their formalism. Our method proposes the use of first order logic including processes. The ability to represent processes is paramount in

legal requirements; hence we find that our work has the advantage of representing processes.

9.8 Summary

The study in this chapter demonstrated the feasibility of employing the GAM as an analysis method with reasonable success to capture and check consistency of governance requirements. GAT was able to generate and check consistency and scenarios within bounded scope for small to medium complexity problems.

The novelty of this domain and the apparent non-existence of comparable solutions do not allow us to compare our tool and method to others. Our proof-of-concept implementation of the examples using our tool provides a modest proof of the ability to validate consistency and partial completeness to governance laws.

The benchmark tests validate the feasibility of our method of representing the governance requirements problem in the form of first order logic and solving the problem using logic analysers dependent on SAT solvers. Most importantly, it shows that it is possible to validate small and medium sized simulations of governance requirements.

The limitations of the SAT solvers to check models with large numbers of instances can be a challenge to checking case studies with large number of instances. However, based on our experience, we can conjecture that violations

or inconsistencies do not require a large number of instances for them to be detected. This chapter has presented various tests applied to the analyser model.

In practice, execution of the case studies that we have presented in Chapters 7 and 8 was very quick, at most about five seconds.

We have provided an analysis of the challenges outlined in the thesis and a description of our approach towards mitigating these challenges. We have also put forward a discussion of comparison with related work. Answered questions of dealing with change, addressing differences in terminology, and required skills for learning the tool.

The next chapter will present a summary of this dissertation along with conclusions and discussions of future work.

Chapter 10

Concluding Remarks

10.0 Introduction

The future of Law, as well as developments in Information Technology, indicate a possibility of involving computers in validating compliance, checking the logic, and possibly defining laws[Walter 1988]. The compliance validation of a governance system requires a check of consistency and completeness. This thesis studies the feasibility of using computer languages and logical methods to assist in the validation of enterprise governance systems. Particularly we address the question of consistency checks by using formal analysis. The ability of a formal model to accurately capture legal requirements is a major challenge. To address this challenge, this thesis proposes a framework that includes a method, meta models, a language and a tool. The meta models and extraction patterns assist in requirements capturing. The models and corresponding patterns were shown to be useful in extracting governance requirements. The language and the tool are used to map extracted requirements into corresponding logic representation. The logic representation follows a first order logic model that mirrors the extraction model. By study of examples we were able to find the overall method and particularly the logic representation to be effective in checking consistency and partial completeness.

This chapter presents concluding remarks for this thesis including a summary, an analysis of the hypothesis statement, a reminder of discussions presented in each of the chapters, a summary of contributions including experiences, primary and secondary contributions, and limitations. The final section presents future work.

Section 10.1 in this chapter will summarize important points presented in the thesis. In section 10.2 we analyse the thesis hypothesis. Section 10.3 provides a chapter-by-chapter synopsis. Section 10.4 shows a summary of contributions including a list of demonstrated summaries, in addition to primary and secondary contributions, and finally discussion of future work. Section 10.5 presents the conclusion.

10.1 Summary

This thesis describes a framework for validating the enterprise legal compliance process. The framework includes legal and enterprise models, a governance analysis method, a governance analysis language, and an implemented prototype of the governance analysis tool.

While developing the framework, we followed several approaches for researching each of the framework components. The main hypothesis was confirmed through our ability to test a number of different examples (see section 10.2).

Our research method depended on a systematic approach to extract models from actual legal examples. Based on examples taken both from law and enterprise control systems, we defined the models and their corresponding patterns.

Our prototype tool was designed and tested to be deterministic. We implemented the tests using several black-box tests written in Python[Python 2009].

Our method to check for legal consistency is threefold: We have model consistency checks, ontology checks, and scenario checks. The three checks were performed using sample input from laws and enterprise requirements. The laws studied were PIPEDA Sections 4.1 and 4.2, in addition to excerpts from SOX-404, 52-109, and 52-111.

Further system validation of the full framework was sought by applying the end-to-end process: extract, represent, and generate processes. For each of the examples we applied the three checks, if applicable. The case studies intended to test the end-to-end implementation of the framework. We were able to assist in checking consistency using each of the techniques.

Our prototype tool was designed and tested to be deterministic. We implemented the test procedures using several black-box tests written in Python.

10.2 Analysing the hypothesis

There are two main conjectures in the hypothesis presented in Chapter 1 and which we have tested in this thesis:

Conjecture 1: It is possible to partially automate the process of compliance validation of enterprise requirements to legal requirements through the use of logic-based models.

Conjecture 2: Logic analysers are useful tools for this purpose.

In the next two subsections, we will list the elements justifying the conjectures and show their locations in the thesis.

10.2.1 Conjecture-1

The first conjecture has some requirements and makes some assumptions. These are listed in the next subsection.

Assumptions and Requirements

This conjecture requires a formal definition of compliance validation. In addition, it assumes that requirements are already written in a language understood by computers. The requirements and assumptions of Conjecture 1 are shown in figure 10.1.

Figure 10.1 shows that the first conjecture has a requirement of defining compliance semantics formally, in addition it assumes that a requirements language and translating mechanisms from requirements to the language exists.

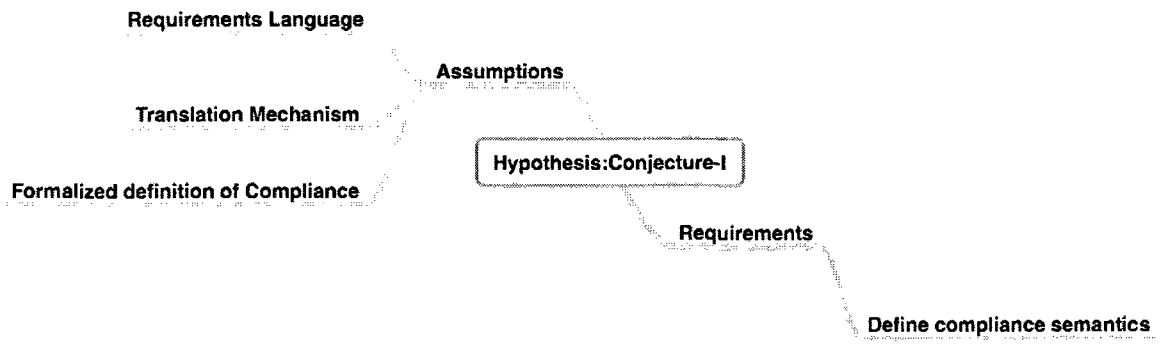


Figure 10.1: Conjecture-I Requirements and Assumptions

Solutions

To address the above-mentioned requirements and assumptions, we have:

- provided semi formal and formal definitions of compliance validation, consistency, compliance and completeness in Chapters 1,2 and 4.
- created an extraction model and method able to reasonably capture legal and enterprise requirements in Chapter 5.
- created a language to represent requirements in Chapter 6.

The language we propose:

- has the ability to reasonably capture the law;
- is easily translated into our logic model
- creates a logic analyzer model definition that corresponds to the combined legal and enterprise requirements model

Figure 10.2 shows a listing of elements of our process of justifying the validation of the conjecture. The Figure lists the three elements that were used: the language, the formal definitions, and the extraction process. In Chapter 4, we showed how the components of our prototype can translate requirements to an analyser model. In Chapter 5, we showed how the requirements can be extracted from the law. In Chapter 6, we showed how we transformed these requirements into a logic representation.

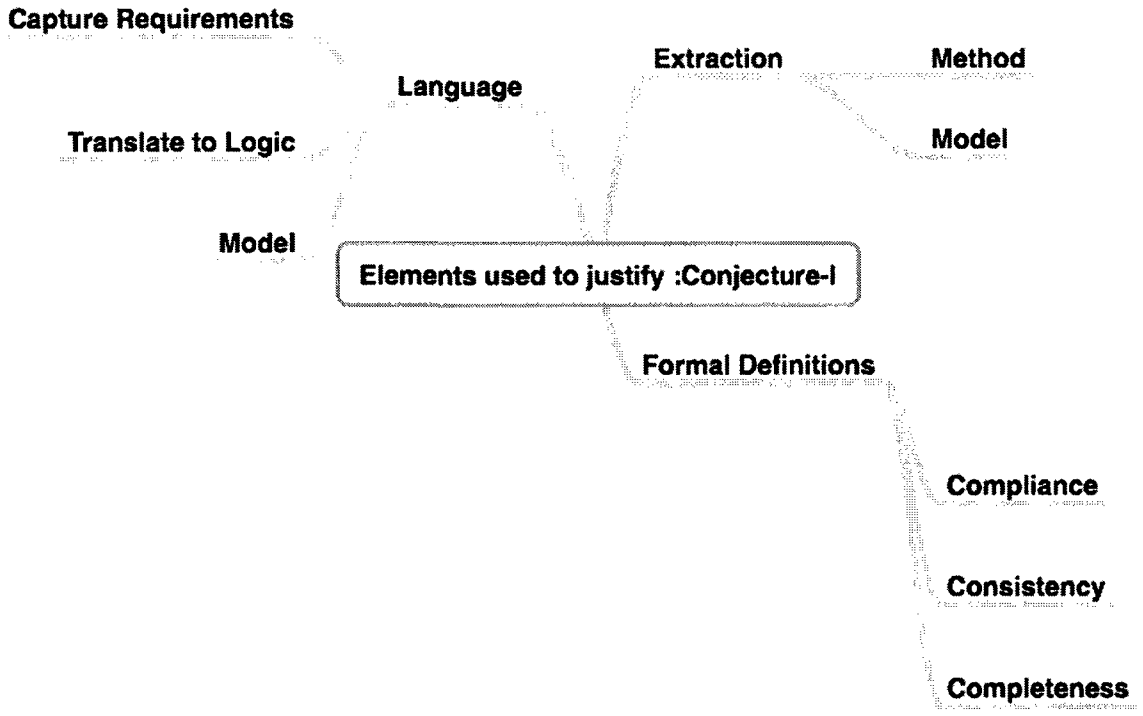


Figure 10.2: Elements used to justify conjecture-I

The prototype was tested using many examples including the ones presented in this thesis, particularly the case studies in Chapters 7 and 8. We provided

informal definitions in Chapters 1 and 2. In addition, we have provided a formal definitions in Chapter 4.

10.2.2 Conjecture-2

Similarly, the second conjecture assumes the ability to produce validation results by using formal requirement validation techniques on the case studies taken from the privacy and security laws.

Requirements and assumptions

The second conjecture of our hypothesis requires techniques able to validate consistency. In addition, it requires showing the ability of the proposed methods to address partial compliance (consistency) by studying governance laws.

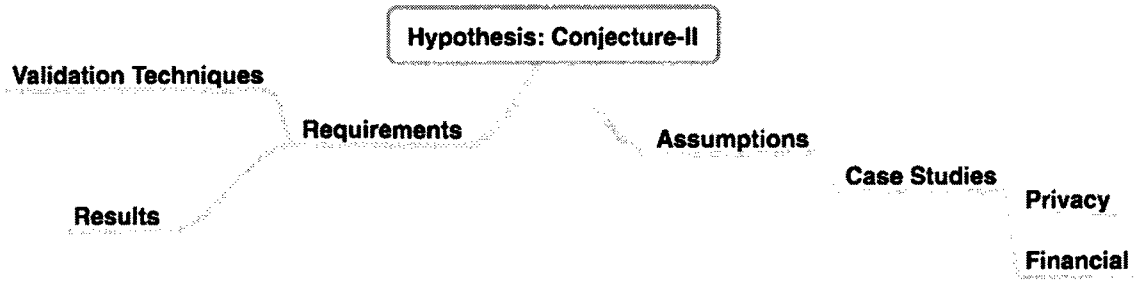


Figure 10.3: Hypothesis conjecture-II

Elements used to justify conjecture-II

To support the second conjecture we have proposed three techniques (also presented in Section 2.2) : model consistency, ontology, and scenario checks.

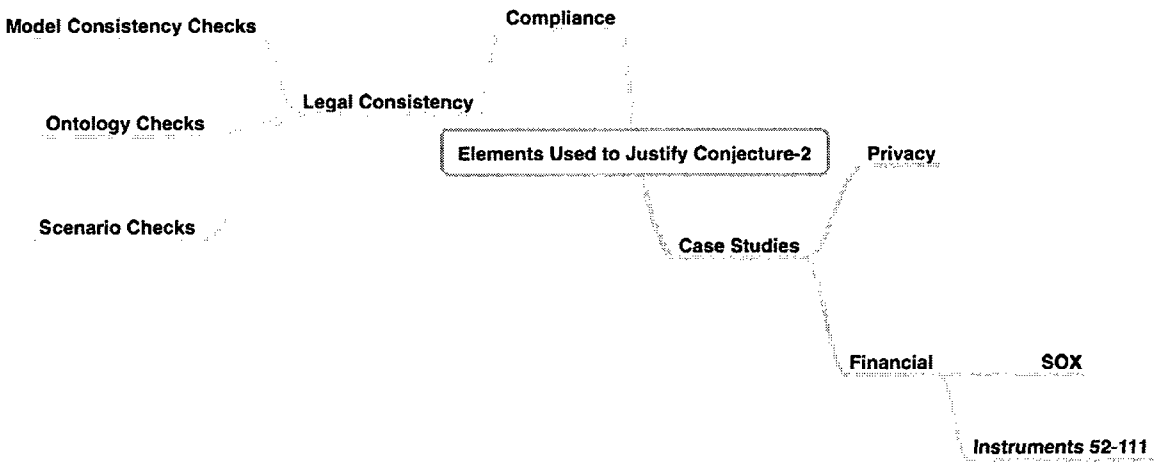


Figure 10.4: Elements used to justify conjecture-II

Chapters 6 and 7 presented an end-to-end validation of the method producing results of consistency checking of requirements from PIPEDA, SOX, and Instruments 52-111 onto various enterprise requirements. Figure 10.4 shows a map of the above mentioned elements.

10.3 Summary of Contributions

The principal contribution and focus of this work is the introduction of the governance analysis framework. Figure 10.5 presents a set of contributions of this thesis. Illustrated on the right are the major components of the framework namely the method, the tool, and the language. Illustrated on the left are the supporting artefacts such as the meta models, the formal representation, the

requirement patterns, the pattern to GAL translation and the GAL to logic mapping.

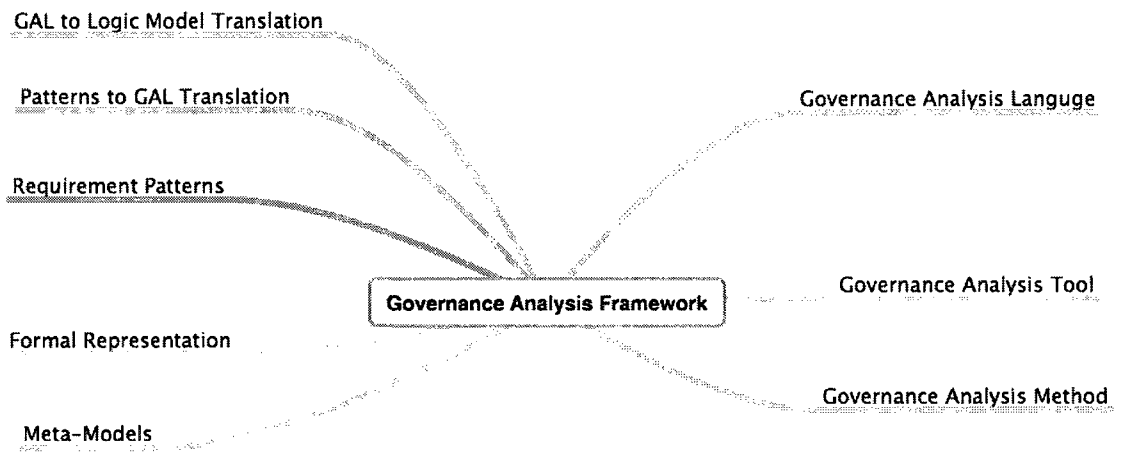


Figure 10.5: Contributions Summary

Research for the method has included a study of the existing literature of requirements extraction and logic checking of requirements originating in laws. In addition, we studied the ability of the model to represent legal and enterprise requirements conceptually through models. Further, we proposed a combined requirements law and enterprise model. We also created a formal representation of these models using logic. The semi-formal UML requirement model can be mapped into a formal logic representation. We studied requirements by analysing various governance laws and summarized nine extraction patterns that reasonably represent legal requirements, at least in our

case studies. We connected the extraction patterns with GAL language statements. The GAL statements were researched to partially satisfy governance requirements. We provided a mapping from patterns to GAL. Finally, we implemented a tool prototype capable of processing GAL statement and producing logic models for analysis. The tool's three main components are the import module, the logic generator module and the results analysis filter.

The GAT prototype has been implemented using python code under the OS X Leopard OS. The GAT wraps the functions of the Alloy-4 analyser, a first-order language and logic-checking tool. However other languages or systems could have been used as well.

Finally we provided case studies that showcased the method and the tools capabilities. We concentrated on repeatability and simplicity factors. The following three subsections outline the experiences demonstrated in this body of work, the primary contributions of this thesis, and the limitations of the method.

10.3.1 Demonstrated Experiences

The experiences reported in this thesis demonstrate that:

- It is possible to extract legal and enterprise requirements using the extraction model and its related patterns.
- It is possible to translate extracted patterns into a set of GAL statements that can then be converted into logic-based language for analysis.

- Examples confirm that model consistency, ontology, and scenario checks are able to detect interactions and sample violations of the law.
- Formal logic analysers offer the ability to represent and check consistency of requirements originating from the law or the enterprise.

10.3.2 Primary Contributions

The primary contributions of this thesis are:

- The governance analysis framework, which includes the meta models, the method, along with a language and tools to validate compliance through consistency checking.
- The Governance Analysis Method, which describes the process of extracting, translating, and validating requirements.
- An extraction model along with its patterns that are used in the requirements extraction processes.
- A formal (logic) representation of the requirements model in addition to a formal representation of the consistency validation techniques suggested.
- Three consistency checks techniques, namely model consistency, ontology, and scenario checks to assist in validating legal consistency of enterprise to legal requirements.

10.3.3 Minor contributions

- A language with defined semantics to represent requirements
- A tool prototype written in Python and implemented under OS X to translate requirements
- A reference model for enterprise requirements
- A legal extraction model for requirements

10.3.4 Limitations

The main limitations of the framework are:

- The method does not include a study of completeness
- The method is dependent on the quality of the requirements as translated by a human agent (Governance Officer), where multiple interpretations may be present
- The lack of existing research into defining semantics of consistency, completeness and compliance
- The method has only been tested on Alloy as a logic analyser, however, it may be possible to test it using other logic analysis tools.
- The translation of declarative and composite requirements requires human effort.

- The extraction method is limited to the capabilities of what the models offer. Therefore, studying the relationship with other extraction methods is also missing from this work.
- The work does not include a tool to assist in requirement extraction.
- The traceability of design requirements to implemented language is missing and hence this is left for future work.
- The method was demonstrated on very specific types of laws, essentially specific portions of two laws only.

10.4 Future Work

The work in this thesis addresses some of the fundamental problems with validating enterprise legal consistency for the sake of validating compliance. This section provides an overview of areas of future interest:

- Addressing the completeness issue of the compliance validation problem, as presented in Chapters 1 and 2. Completeness is an important aspect of enterprise compliance. If enterprise requirements are incomplete with respect to the law, then the enterprise is non-compliant.
- Applying the method on other governance laws.
- Creating a knowledge base of composite patterns, in order to facilitate the requirements extraction process, which will be executed by the GO.

- Study comparatively GRL and GAM.
- Implementing a tool that assists users in extracting legal requirements by way of matching them with patterns.
- Adding a module to GAT to read BPEL representations. This would simplify the task of the governance officer. A GO would be able to use visual mechanisms to describe the enterprise. Once complete, such descriptions can be imported by GAT to analyse updated models
- Writing a GAT module able to read access control policies represented in XACML or any other policy representation language
- Writing a UCM to GAL translator that allows a UCM description to be written in GAL then translated into Alloy.
- Integrate a traceability tool that connects the legal and enterprise requirements text into the logic model.
- Creating an XML schema for GAL.

Along with some of the examples listed above, there are many other possibilities for improving various aspects of this work and expanding it to provide improved compliance validation mechanisms.

10.5 Conclusion

The methodology proposed in this thesis was made possible by the models and architectures presented in Chapter 2, as well as by the efficient and flexible logic analyzer Alloy that we have used. Combining requirements and transforming them into a logic analyser model presents a novel approach in the area of compliance, particularly for consistency validation. The difficulty of this work was to map a complex problem into simpler terms for computing purposes.

Our method needed early validation while under development. The initial case studies we implemented focused on small examples and the ability to detect model inconsistencies. The larger examples concentrated on validating scenarios. We have found that the major difficulty in implementing laws is the ability to see the hidden effects simple provisions may have. However, our chosen model analyser (Alloy) is known for its ability to uncover otherwise unnoticed scenarios by generating instances.

Validation of the Governance Analysis Method (GAM) involved three major steps:

The GAM was used to extract requirements for compliance, followed by the construction of a prototype that translated extracted requirements (Chapters 5 and 6).

Several small examples in addition to two case studies were implemented, involving privacy and financial requirements. These examples allowed us to evaluate the suitability of the method for governance laws (Chapters 7 and 8).

Performance tests were run, measuring the generated model's ability to provide solutions for models with complex logic formulas, possibly including complex ontologies (Chapter 9).

While differing in research approach, each of steps sought to validate the method.

Appendix

Privacy Legistlation Across Canada

1. Alberta: Freedom of Information and Protection of Privacy Act, R.S.A. 2000,c. F. 25
2. British Columbia: Freedom of Information and Protection of Privacy Act R.S.B.C. 1996, c. 165
3. Federal: *Privacy Act*, R.S.C., c.P-21.
4. Manitoba: The Freedom of Information and Protection of Privacy Act C.C.S.M. F175
5. New Brunswick: Protection of Personal Information Act S.N.B. 1998, c. P19.1
6. Newfoundland and Labrador: Access to Information and Protection of Privacy *Act* S.N.L. 2002, can-1.1122
7. Northwest Territories: Access to Information and Protection of Privacy Act, S.N.W.T. 1994.
8. Nova Scotia: Freedom of Information and Protection of Privacy Act, R.S.N.S. 1993.
9. Nunavut: Access to Information and Protection of Privacy Act, S.N.W.T. 1994.
10. Ontario: Freedom of Information and Protection of Privacy Act, R.S.O. 1990.
11. Ontario: Municipal Freedom of Information and Protection of Privacy Act R.S.O. 1990.
12. Prince Edward Island: Freedom of Information and Protection of Privacy Act R.S.P.E.I. 1998.

13. Quebec: An Act respecting Access to documents held by public bodies and the Protection of personal information, R.S.Q.

14. Saskatchewan: Freedom of Information and Protection of Privacy Act, S.S. 1990.

15. Saskatchewan: Local Authority Freedom of Information and Protection of *Privacy Act*, S.S. 1990-91.

16. Yukon: Access to Information and Protection of Privacy Act, R.S.Y. 2002.

Index

References

[Agotnes 2007] Agotnes, T. W. Van der Hoek, J. A. Rodriguez-Aguilar, C. Sierra, and M. Wooldridge. On the logic of normative systems. In Proc. IJCAI-07, Hyderabad, India, 2007.

[Alchourròn 1971] Alchourròn, C.E., Bulygin, E.: Normative Systems. Springer, 1971.

[Amyot 2001] Amyot, D. 2001 Specification and Validation of Telecommunications Systems with Use Case Maps and Lotos. Doctoral Thesis.

[Anand 2007] Essentials of Sarbanes-Oxley By Sanjay Anand Published by John Wiley and Sons, 2007 ISBN 0470056681, 9780470056684.

[Antón 2007] Antón, A. I., Bertino, E., Li, N., and Yu, T. 2007. A roadmap for comprehensive online privacy policy management. Commun. ACM 50, 7 (Jul. 2007), 109–-116. DOI= http://doi.acm.org/10.1145/1272516.1272522.

[Armour 1999] Armour, J. Share Capital and Creditor Protection: Efficient Rules for a Modern Company Law. ESRC Centre for Business Research - Working Papers wp148, ESRC Centre for Business Research. 1999.

[Arter 2002] Dennis R. Arter Published by American Society for Qualit, 2002 ISBN 873895703, 9780873895705.

[Ashley 2004] Ashley , P. A privacy logging and reporting framework. In Proceedings of The Asia Pacific Information Technology Security Conference (University ofQueensland, 24–27 May 24–27, 2004).

[Ayers 1992] Ayres, Ian and Gertner, Robert (1992), 'Strategic Contractual Inefficiency and the Optimal Choice of Legal Rules', 101 Yale Law Journal 729

[Bailey 1995] The Encyclopaedia of Police Science By William G. Bailey Edition: 2, illustrated Published by Taylor & Francis, 1995 ISBN 0815313314, 9780815313311

[Balust 2001] Balust, J. M. And Franch, X. 2001. Building Expressive and Flexible Process Models Using a UML-Based Approach. In Proceedings of the 8th European Workshop on Software Process Technology (19–21 June 19 - 21, 2001). V. Ambriola, d. Lecture Notes In Computer Science, vol. 2077. Springer-Verlag, London, 152–-172.

[Barth 2006] Barth A., Datta A., Mitchell J., Nissenbaum H.: Privacy and Contextual Integrity: Framework and Applications. IEEE Symposium on Security and Privacy 2006: 184-198

[Belarus 1997] Belarus: Prices, Markets, and Enterprise Reform By World Bank Edition: illustrated Published by World Bank Publications, 1997 ISBN 0821339761, 9780821339763

[Bench-Capon 2008] Bench-Capon T. , Prakken H. : Introducing the Logic and Law Corner. J. Log. Comput. 18(1): 1-12 (2008).

[Benditt 1978] Law as Rule and Principle: Problems of Legal Philosophy By Theodore M.

Benditt Published by Stanford University Press, 1978 ISBN 0804709637, 9780804709637

[Bhatnagar 2004] E-government: From Vision to Implementation : a Practical Guide with Case Studies By Subhash C. Bhatnagar Edition: illustrated Published by Sage Publications, 2004 ISBN 0761932593, 9780761932598.

[Biagioli 2005] Biagioli, C., Francesconi, E., Passerini, A., Montemagni, S., and Soria, C. 2005. Automatic semantics extraction in law documents. In Proceedings of the 10th international Conference on Artificial intelligence and Law (Bologna, Italy, June 06 - 11, 2005). ICAIL '05. ACM, New York, NY, 133-140. DOI= http://doi.acm.org/10.1145/1165485.1165506

[Boella 2006] Guido Boella and Leendert van der Torre. A Logical Architecture of a Normative System. L. Goble and J.-J.C. Meyer (Eds.): DEON 2006, LNAI 4048, pp. 24–35, 2006. © Springer-Verlag Berlin Heidelberg 2006

[Bourcier 2003] Legal Knowledge and Information Systems: JURIX 2003 : the Sixteenth Annual Conference By Danièle Bourcier, Foundation for Legal Knowledge Systems. Published by IOS Press, 2004 ISBN 1586033980, 9781586033989.

[Bown 2009] J. Bowen. Formal Methods. http://www.afm.lsbu.ac.uk.

[Braganza 2007] Braganza, A. And Franken, A. 2007. SOX, compliance, and power relationships. Commun. ACM 50, 9 (Sep. 2007), 97—102. DOI= http://doi.acm.org/10.1145/1284621.1284626.

[Breaux 2006] Breaux, T. D., Vail, M. W., and Antón, A. I. 2006. Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations. In Proceedings of the 14th IEEE international Requirements Engineering Conference (Re'06) 11–15 S (September 11 - 15, 2006). RE. IEEE Computer Society, Washington, DC, 46—55. DOI= http://dx.doi.org/10.1109/RE.2006.68http://dx.doi.org/10.1109/RE.2006.68.

[Breaux 2008] Breaux , T. And Antón, A. 2008. Analyzing Regulatory Rules for Privacy and Security Requirements. IEEE Trans. Software Eng. 34, 1 (Jan. 2008), 5—20. DOI=Http://dx.doi.org/10.1109/TSE.2007.70746.

[Breuker 2004] J. Breuker, A. Valente, and R. Winkels. Legal ontologies in knowledge engineering and information management. Artificial Intelligence and Law, 12(4):241–277, 2004.

[Brodie 2006] Brodie, C. A., Karat, C., and Karat, J. : An empirical study of natural language parsing of privacy policy rules using the SPARCLE policy workbench. In SOUPS '06, ACM Internat. Conf. Proc. Series Vol. 149, 8-19.

[Cannon 2006] Cannon, J. C. And Byers, M. 2006. Compliance deconstructed. Queue 4, 7 (Sep. 2006), 30-37. DOI= http://doi.acm.org/10.1145/1160434.1160449.

[Capdevila 2005] Chris Capdevila , Enforcing Application Controls under SOX: A Closed-Loop Approach, SOX compliance journal, July 13 2005.

[CDT 2009] http://www.cdt.org/privacy/eudirective/E_Directive.html , Accessed Nov 2008.

[Chomicki 2003] Chomicki, J., Lobo, J., and Naqvi, S. 2003. Conflict Resolution Using Logic Programming. IEEE Trans. On Knowl. And Data Eng. 15, 1 (Jan. 2003), 244—249.

[Chung 2000] Chung, L., Nixon, B. A, Yu, E., and Mylopoulos, J: Non-Functional Requirements in Software Engineering Kluwer Academic Publishers, Dordrecht, USA, 2000.

[Cohn 2007] Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000), Breckenridge, Colorado, April 12-15 2000 By A. G. Cohn, Fausto Giunchiglia, Bart Selman Published by Morgan Kaufmann Publishers, 2000 Original from the University of Michigan Digitized Nov 2, 2007 ISBN 1558606904, 9781558606906

[Colby 1987] The Measurement of Moral Judgment By Anne Colby, Lawrence Kohlberg Contributor Lawrence Kohlberg Edition: illustrated Published by CUP Archive, 1987 ISBN 0521325013, 9780521325011

[Cornelius 2006] OMB Circular A-123 and Sarbanes-Oxley: Management's Responsibility for Internal Control in Federal Agencies By Cornelius E. Tierney, Cornelius E Tierney, CPA, Edward F. Kearney, Roldan Fernandez, Jeffrey W. Green, Michael J. Ramos, Kearney & Company, Kearney & Company Published by For Dummies, 2006 ISBN 0470036893, 9780470036891.

[CSA 1996] Model Code for the Protection of Personal Information. Tech. Rep., Canadian Standards Association. Http://www.csa.ca/standards/privacy/code/.

[Dalton 2008] Dalton, D., Dalton, C., Corporate governance in the post Sarbanes-Oxley period: Compensation disclosure and analysis (CD&A), Business horizonsvolume 51, Issue 2., March–April 2008, pp. 85–92. DOI=Http://doi:10.1016/j.bushor.2007.10.004.

[Darimont 1997] Darimont, R., Delor, E., Massonet, P., and van Lamsweerde, A. 1997. GRAIL/KAOS: an An environment for goal-driven requirements engineering. In Proceedings of the 19th international Conference on Software Engineering (Boston, Massachusetts, United States, 17–23 May 17 - 23, 1997). ICSE '97. ACM, New York, NY, 612-613. DOI= http://doi.acm.org/10.1145/253228.253499DOI= http://doi.acm.org/10.1145/253228.253499.

[Deek 2005] Strategic Software Engineering: An Interdisciplinary Approach By Fadi P. Deek, James A. McHugh, Osama M. Eljabiri Edition: illustrated Published by CRC Press, 2005 ISBN 0849339391, 9780849339394

[Dignam 2006] Company Law By Alan Dignam, John Lowry Published by Oxford University Press, 2006 ISBN 0199289360, 9780199289363.

[Dubois 1994] Dubois, E. 1994. Use of deontic logic in the requirements engineering of composite systems. In Deontic Logic in Computer Science: Normative System Specification, J. C. Meyer and R. J. Wieringa, Eds. John Wiley & Sons, New York, NY, 125—139.

[Evers 2008] An analysis of the requirements for DSS on integrated river basin management. Mariele Evers. Management of Environmental Quality: An International Journal. 2008. Vol: 19 Issue: 1 Page:p 37–53. DOI: 10.1108/14777830810840354. Emerald Group Publishing

Limited.

[Fagin 2003] Reasoning about Knowledge By Ronald Fagin, Joseph Y. Halpern, Yoram Moses, Moshe Y. Vardi Edition: illustrated Published by MIT Press, 2003 ISBN 0262562006, 9780262562003.

[Fajardo 2007] Testing SAP R/3: A Manager's Step-by-Step Guide By Jose Fajardo, Elfriede Dustin, Inc netlibrary Published by Wiley-Interscience, 2007 ISBN 0470135484, 9780470135488.

[Fischer-Hubner 1998] Fischer-Hubner, S. And Ott, A. From a Formal Privacy Model to its Implementation. Proceedings of the 21st National Information Systems Security Conference, Arlington, VA, 5–8 October 5-8, 1998.

[Floridi 2007] The Blackwell Guide to the Philosophy of Computing and Information By Luciano Floridi Published by Blackwell Publishing, 2007 ISBN 0631229191, 9780631229193

[Fox 2006] IT Control Objectives for Sarbanes-Oxley: The Role of It in the Design and Implementation of Internal Control Over Financial Reporting By IT Governance Institute, Christopher Fox, Paul Zonneveld, IT Governance Institute Edition: 2, illustrated Published by ISACA, 2006 ISBN 1933284765, 9781933284767

[Gamut 1991] Logic, Language, and Meaning: Introduction to Logic By L. T. F. Gamut. Edition: illustrated Published by University of Chicago Press, 1991

[Gavit 1952] Cases and Materials on an Introduction to Law and the Judicial Process By Bernard C. Gavit, Ralph F. Fuchs, Monrad G. Paulsen Edition: 2 Published by Callaghan, 1952 Original from the University of California

[GB 2004] Business Perspective: The IS View on Delivering Services to the Business By Great Britain. Office of Government Commerce, Great Britain Office of Government Commerce, Office of Government Commerce Edition: 2, illustrated Published by The Stationery Office, 2004 ISBN 0113308949, 9780113308941

[George 2003] Computers in Society: Privacy, Ethics, and the Internet By Joey F. George Contributor Joey F. George Edition: illustrated Published by Pearson Prentice Hall, 2003 Original from the University of Michigan Digitized Nov 19, 2007 ISBN 0131406604, 9780131406605

[Ghanavati 2007] S. Ghanavati, D. Amyot, and L. Peyton. Towards a framework for tracking legal compliance in healthcare. In J. Krogstie, A. L. Opdahl, and G. Sindre, editors, 19th International Conference on Advanced Information Systems Engineering (CAiSE'07), volume , pages 218–232. Springer, 2007.

[Ghanavati 2008]Ghanavati, S., Amyot D., and Peyton, L. (2008) Comparative Analysis between Document-based and Model-based Compliance Management Approaches. First Int. Workshop on Requirements Engineering and Law (RELAW 2008), Barcelona, Spain, September. IEEE, 35-39.

[Ghanavati 2009] Ghanavati, S., Amyot, D., and Peyton, L. (2009) Compliance Analysis Based

on a Goal-oriented Requirement Language Evaluation Methodology. 17th IEEE International Requirements Engineering Conference (RE'09), Atlanta, USA, September 2009. IEEE CS.

[Giblin 2005] Giblin, A. Y. Liu, S. Müller, B. Pfitzmann, and X. Zhou, "Regulations Expressed as Logical Models (REALM)," Proceedings of the 18th Annual Conference on Legal Knowledge and Information Systems, Brussels, Belgium (2005), pp. 37-48.

[Goergen 2007] Goergen , M., Manjon, M. C., & Renneboog, L., Recent Developments in German Corporate Governance, International Review of Law & Economics (2007), doi:10.1016/j.irle.2008.06.003

[Graham 2001] Statutory Interpretation: Theory and Practice By Randy N. Graham, Canada. Published by Emond Montgomery Publication, 2001 ISBN 1552390632, 978155239063.

[Gupta 2008] Anil Gupta, URL http://www.metricstream.com/insights/sox_it_controls.htm, Accessed Dec 2008.

[Hambrick 2008] Hambrick, C., Werder, Edward J. Zajac New Directions in Corporate Governance Research organizationscience. Vol. 19, No. 3, May–June 2008, pp. 381–385. Doi 10.1287/orsc.1080.0361.

[Hassan 2005] Hassan, W. , Logrippo L.: Governance Policies for Privacy Access Control and their Interactions. Feature Interaction Workshop 2005: 114-130.

[Hassan 2006-1] Hassan, W. Process Based Access Control. Technical Report UofO. School of engineering and information technology-University of Ottawa Seminar. Jan 2006.

[Hassan 2006-4] Hassan, W. Subject Verb Object SVO model for access control. Technical Report UofO. School of engineering and information technology-University of Ottawa Seminar. March 2006.

[Hassan 2007] Hassan, W. Reduced Policy Instruction Set. Technical Report UofO. School of engineering and information technology-University of Ottawa Seminar. January 2007.

[Hassan 2008-1] Hassan, W. Process-based Privacy Governance Model for Governments and Enterprises. Technical Report UofO. School of engineering and information technology-University of Ottawa Seminar. Jan 2008.

[Hassan 2008-4] Hassan, W., April 1st, 2008, Formal Privacy Ontology: A Definition, ICSTC Computer Science track, San Diego.

[Hassan 2008-9] Hassan, W., Logrippo, L. Requirements and Compliance in Legal Systems: a Logic Approach. In Proc. IEEE 16th International Requirements Engineering Conference (RE'08): RELAW Workshop. Barcelona, Spain. Sep. 2008

[Hassan 2009] Hassan, W., Logrippo, L.: Validating Compliance with Privacy Legislation. Submitted for publication.

[Hruby 2006] Hruby, P., Kiehn, J., Scheller, C. V. (2006). Model-Driven Design Using

Business Patterns. Springer. ISBN 3-540-30154-2.

[Hugh 2006] Sustained SOX: A Practical Guide for Implementing a Sustainable Sarbanes Oxley Process By Michael S. Hugh Published by AuthorHouse, 2006. ISBN 1425924832, 9781425924836

[IFRS 2009] http://www.iasb.org/IFRS+Summar. Accessed Jan 2009.

[INST 52-109] Multilateral Instrument 52—109 Certification of Disclosure in Issuers' Annual and Interim Filings.

[ISTAR 2009] i* Framework, http://www.cs.toronto.edu/km/istar/, Accessed Jan 2009.

[ITUT 2003] ITU-T: User Requirements Notation (URN) – Language Requirements and Framework. ITU-T Recommendation Z.150. Geneva, Switzerland, February 2003.

[Jackson 2003] Jackson, D., Al-Kofahi, K., Tyrrell, A., and Vachher, A. 2003. Information extraction from case law and retrieval of prior cases. Artif. Intell. 150, 1-2 (Nov. 2003), 239-290. DOI= http://dx.doi.org/10.1016/S0004-3702(03)00106-1.

[Jackson 2006] Jackson D. Software Abstractions: Logic, Language, and Analysis. MIT Press. Cambridge, MA. March 2006. ISBN 0-262-10114-9. Proceedings of the 1st IEEE International Symposium on Requirements Engineering, pages 56–64.

[jajodia ed. 2008] Proceedings of the IFIP TC 11 23rd International Information Security Conference: IFIP 20th World Computer Congress, IFIP SEC'08, 7–10 September 2008, Milano, Italy edited by Sushil Jajodia, Pierangela Samarati, Stelvio Cimato Contributor Sushil Jajodia, Pierangela Samarati, Stelvio Cimato Published by Springer, 2008 ISBN 0387096981, 9780387096988.

[Jarke 1998] Jarke, M. & Kurki-Suonio, R. (1998). Guest Editorial - – Special issue on Scenario Management. 1EEE Transactions on Software Engineering, 24(12).

[Johns 1911] The Code of Laws Promulgated by Hammurabi, King of Babylon B.C. 2285 - 2242 By Hammurabi, C H W Johns Published by T.& T. Clark, 1911 Original from the University of California Digitized Nov 15, 2007

[Jones 1994] Jones, A. J. And Sergot, M. 1994. On the characterization of law and computer systems: the normative systems perspective. In Deontic Logic in Computer Science: Normative System Specification, J. C. Meyer and R. J. Wieringa, Eds. John Wiley & Sons, New York, NY, 275-307.

[Kelsen 1945] Kelsen, H.: General Theory of Law and State. Harvard University Press, 1945.

[Kerrigan 2003] Kerrigan, S. And Law, K. H. 2003. Logic-based regulation compliance-assistance. In Proceedings of the 9th international Conference on Artificial intelligence and Law (Scotland, United Kingdom, June 24 - 28, 2003). ICAIL '03. ACM, New York, NY, 126-135. DOI= http://doi.acm.org/10.1145/1047788.1047820

[Kiyavitskaya 2007] Kiyavitskaya, N., Zeni, N., Breaux, T. D., Antón, A. I., Cordy, J. R., Mich,

L., and Mylopoulos, J. 2007. Extracting rights and obligations from regulations: toward a toolsupportetool supported Process. In Proceedings of the Twenty-Second IEEE/ACM international Conference on Automated Software Engineering (Atlanta, Georgia, USA, November 05 - 09, 2007). ASE '07. ACM, New York, NY, 429-432. DOI= http://doi.acm.org/10.1145/1321631.1321701.

[Kontogiannis 2007] Kontogiannis, K., Lewis, G. A., Smith, D. B., Litoiu, M., Muller, H., Schuster, S., and Stroulia, E. 2007. The Landscape of Service-Oriented Systems: A Research Perspective. In *Proceedings of the international Workshop on Systems Development in SOA Environments* (20–26 May 20 - 26, 2007). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 1. DOI= http://dx.doi.org/10.1109/SDSOA.2007.1. 2

[Kowalski 1985] Kowalski R., Sergot M. Computer Representation of the Law. IJCAI 1985: 1269-1270.

[Kudo 2007] Kudo, M., Araki, Y., Nomiyama, H., Saito, S., and Sohda, Y. 2007. Best practices And tools for personal information compliance management. IBM Syst. J. 46, 2 (Apr. 2007), 235–253.

[Lanni 2006] Law and justice in the courts of classical Athens By Adriaan Lanni Published by Cambridge University Press, 2006 ISBN 0521857597, 9780521857598

[Lee 2005] Lee, S. W. and Gandhi, R. A. 2005. Ontology-based Active Requirements Engineering Framework. In Proceedings of the 12th Asia-Pacific Software Engineering Conference (15–17 December 15 - 17, 2005). APSEC. IEEE Computer Society, Washington, DC, 481—490. DOI= http://dx.doi.org/10.1109/APSEC.2005.86 DOI= http://dx.doi.org/10.1109/APSEC.2005.86.

[Lee 2006] Lee, S., Gandhi, R., Muthurajan, D., Yavagal, D., and Ahn, G. 2006. Building problem Domain ontology from security requirements in regulatory documents. In Proceedings of the 2006 International Workshop on Software Engineering For Secure Systems (Shanghai, China, 20–21May 20 -21, 2006). SESS '06. ACM, New York, NY, 43-50. DOI= Http://doi.acm.org/10.1145/1137627.1137635.

[Lerner 2000] Middleware Networks: Concept, Design, and Deployment of Internet Infrastructure By Michah Lerner, George Vanecek, Nino Vidovic, Dado Vrsalovic Edition: illustrated Published by Springer, 2000 ISBN 0792378407, 9780792378402

[Li 2003] Li, N., Yu, T., and Antón, A.I. A semantics-based approach to privacy languages. CERIAS Technical Report TR 2003–28. Purdue University, Nov. 2003.

[Li 2008] Conceptual Modeling - Er 2008: 27th International Conference on Conceptual Modeling, Barcelona, Spain, October 20-24, 2008, Proceedings edited by Qing Li, Stefano Spaccapietra, Eric Yu, Antóni Olivé Contributor Qing Li, Ph. Published by Springer, 2008 ISBN 3540878769, 9783540878766

[Liu 2003] Liu, L., Yu, E., and Mylopoulos, J. 2003. Security and Privacy Requirements Analysis within a Social Setting. In Proceedings of the 11th IEEE international Conference

on Requirements Engineering (September 08 - 12, 2003). RE. IEEE Computer Society, Washington, DC, 151.

[Lodder 2008] Legal Knowledge and Information Systems: JURIX 2007: the Twentieth Annual Conference By Arno R. Lodder, Laurens Mommers Contributor Arno R. Lodder Edition: illustrated Published by IOS Press, 2008 ISBN 1586038109, 9781586038106

[Logrippo 2007] Logrippo, L. Normative Systems: the Meeting Point between Jurisprudence And Information Technology In: H. Fujita, D. Pisanelli (Eds.): New Trends in Software Methodologies, Tools and Techniques – Proc. Of the 6th somet_07. IOS Press, 2007, 343-354.

[McCarthy 1982] McCarthy, W. The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. The Accounting Review (July 1982) pp. 554-78

[McCarty 2007] McCarty , L. T. 2007. Deep semantic interpretations of legal texts. In Proceedings of the 11th international Conference on Artificial intelligence and Law (Stanford, California, 04–08 June 04 - 08, 2007). ICAIL '07. ACM, New York, NY, 217—224. DOI= http://doi.acm.org/10.1145/1276318.1276361.

[MiniSat 2006] Eén and N. Sörensson. The MiniSat page, http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/, Accessed 2006

[Mitra 2006] Mitra, P., Pan, C., Liu, P., and Atluri, V. 2006. Privacy-preserving semantic interoperation and access control of heterogeneous databases. In Proceedings of the 2006 ACM Symposium on information, Computer and Communications Security (Taipei, Taiwan, 21–24March 21 - 24, 2006). ASIACCS '06. ACM Press, New York, NY, 66-77. DOI=Http://doi.acm.org/10.1145/1128817.1128831.

[Moses 2007] Moses, T., ed., extensible Access Control Markup Language (XACML), Version 2.0; OASIS Standard, retrieved on 2nd November, 2007 from: Http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-specos.Pdf.

[Moskewicz 2001] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, Chaff: engineering an efficient SAT solver , in Proc. 38th ACM/IEEE Design Automation Conference, pp. 530-535, Las Vegas, Nevada, 2001.

[Narain 2008] Narain, S., Levin, G., Malik, S., and Kaul, V. 2008. Declarative Infrastructure Configuration Synthesis and Debugging. J. Netw. Syst. Manage. 16, 3 (Sep. 2008), 235-258. DOI= http://dx.doi.org/10.1007/s10922-008-9108-y

[Narain 2008] Narain, S., Levin, G., Malik, S., and Kaul, V. 2008. Declarative Infrastructure Configuration Synthesis and Debugging. J. Netw. Syst. Manage. 16, 3 (Sep. 2008), 235-258. DOI= http://dx.doi.org/10.1007/s10922-008-9108-y.

[Needles 1993] Principles of Accounting By Belverd E. Needles, Henry R. Anderson, James C. Caldwell Edition: 5, illustrated Published by Houghton Mifflin Co., 1993 ISBN 0395624843, 9780395624845

[Nettleton 2004] Electronic Record Keeping: Achieving and Maintaining Compliance with 21 CFR Part 11 and 45 CFR Parts 160, 162, and 164 By David Nettleton, Janet Gough. Edition: illustrated Published by Informa Health Care, 2004 ISBN 0849321646, 9780849321641

[Newkirk 1988] Understanding Writing: Ways of Observing, Learning, and Teaching. By Thomas Newkirk, Nancie Atwell Published by Heinemann, 1988 Original from the University of Michigan Digitized Mar 29, 2006 ISBN 0435084410, 9780435084417.

[Nikodemus 2005] Wissens management und Innovation Referenzmodellierung zur Prozessoptimierung im Business-to-business-marketing By Paul Nikodemus Published by Cuvillier Verlag, 2005. ISBN 386537400X, 9783865374004.

[OISPP 2009] http:// www. Oispp.ca. gov/consumer_privacy/laws/,Accessed 2008.

[Onabajo 2009] Onabajo A. Analysis of Multilateral Software Confidentiality Requirements. PhD Thesis. 2009.

[OTA-CIT 1984] Effects of information technology on financial services systems. Washington, D.C: U.S. Congress, Office of Technology Assessment, OTA-CIT-202, September 1984. ISBN 1428923675, 9781428923676

[Otto 2007] Otto Paul N. And Antón Annie I. Addressing Legal Requirements in Requirements Engineering, 15th IEEE International Requirements Engineering Conference, Delhi, India, pp. 5– 4, 15–19 October 2007.

[Peltier 2005 Information Security Risk Analysis By Thomas R. Peltier Edition: 2, illustrated, revised Published by CRC Press, 2005 ISBN 0849333466, 978084933346

[Periorellis 2007] Securing Web Services: Practical Usage of Standards and Specifications

[Peyton 2007] Peyton, L., Ghanavati, S. and Amyot, D. (2007) Designing for Privacy Compliance and Performance Management in Health Care. Design Principles and Practices: An International Journal, Vol. 1, Number 3, Common Ground, 2007, 13-25.

[Pinkal 1993] Semantic Formalisms in Natural Language Processing By Manfred Pinkal, Remko Scha, Lenhart Schubert Published by IBFI, Schloss Dagstuhl, 1993 Original from the University of California

[PIPEDA 1999] Personal Information Protection and Electronic Documents Act (PIPEDA). Second Session. Thirty-sixth Parliament. 48–49 Elizabeth II. 1999–2000.

[Podgórecki 1974] Law and Society By Adam Podgórecki Published by Routledge, 1974 ISBN 0710079834, 9780710079831

[Pourshahid 2007] Pourshahid, A. and Tran, T. 2007. Modeling trust in e-commerce: an approach based on user requirements. In Proceedings of the Ninth international Conference on Electronic Commerce (Minneapolis, MN, USA, August 19 - 22, 2007). ICEC '07, vol. 258. ACM, New York, NY, 413-422. DOI= http://doi.acm.org/10.1145/1282100.1282179.

[Priest 2006] In Contradiction: A Study of the Transconsistent By Graham Priest Published

by Oxford University Press, 2006 ISBN 0199263302, 9780199263301

[Python 2009] Python Website Version 3. http://www.python.org. Accessed Dec 2008.

[Qin 2005] Qin, C. 2005. Personal data protection in electronic business. In Proceedings of the 7th international Conference on Electronic Commerce (Xi'an, China, August 15 - 17, 2005). ICEC '05, vol. 113. ACM, New York, NY, 893-895. DOI= http://doi.acm.org/10.1145/1089551.1089727

[Ramos 2006] How to Comply with Sarbanes-Oxley Section 404: Assessing the Effectiveness of Internal Control By Michael J. Ramos Published by John Wiley and Sons, 2006 ISBN 0471740667, 9780471740667.

[Roessler 2007] Roessler T, Wenning R. Challenges for Privacy Policy Languages. In Proceedings of the W3C Workshop on Languages for Privacy Policy Negotiation and Semantics-Driven Enforcement. Retrieved on 25th September 2007, from http://www.w3.org/2006/07/privacy-ws/papers/29-wenning-position/.

[Roscoe 1994] Roscoe, A. W. 1994. Model-checking CSP. In A Classical Mind: Essays in Honour of C. A. R. Hoare, A. W. Roscoe, Ed. Prentice-Hall International Series In Computer Science. Prentice Hall International (UK) Ltd., Hertfordshire, UK, 353-378.

[Roscoe 1997] Information Technology — Z Formal Specification Notation — Syntax, Type System and Semantics (ISO/IEC 13568:2002 ed.). 2002-07-01. pp. 196 pages. Roscoe, A. W. (1997). The Theory and Practice of Concurrency. Prentice Hall. ISBN 0-13-674409-5.

[SACF 2008] Security, Audit and Control Features Oracle E-Business Suite: A Technical and Risk Management Reference Guide By Deloitte Touche Tohmatsu Research Team and ISACA, Information Systems Audit and Control Association Contributor Information Systems Audit and Control Association Staff, Information Systems Audit and Control Association Edition: 2, illustratedPublished by ISACA, 2008 ISBN 193328451X, 9781933284514

[Sartor 2005] Giovanni Sartor. Legal Reasoning: A Cognitive Approach to the Law. Vol. 5. Treatise on Legal Philosophy and General Jurisprudence. Berlin: Springer, 2005

[Schunter 2007] Schunter , M., Van Herreweghen, E., and Waidner, M. Expressive privacy promises – —How to improve the Platform for Privacy Preferences (P3P). Position paper fr W3C. Workshop on the Future of P3P Retrieved on 30th May 2007 from http://www.w3.org/2002/p3pws/pp/ibm-zuerich.pdf.

[SEC 1934] SECURITIES AND EXCHANGE COMMISSION 17 CFR Parts 232, 240, and 249[Release No. 34–52029; File No. S7–25–04] RIN 3235–AJ04 Removal From Listing and Registration of Securities Pursuant to Section 12(d) of the Securities Exchange Act of 1934.

[Seffah 2005] Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle: Integrating Usability in the Software Development Lifecycle By Ahmed Seffah, Jan Gulliksen, Michel C. Desmarais Edition: illustrated Published by Springer, 2005 ISBN 140204027X, 9781402040276

[Sergot 1982] M.J. Sergot. Prospects for representing the law as logic programs. In K.L. Clark and S.Â. Tarnlund, editors, Logic Programming, pages 33--42. Academic Press, London, 1982

[Shleifer 1996] Shleifer, Andrei and Vishny, Robert W.,A Survey of Corporate Governance(April 1996). NBER Working Paper No. W5554. Available at SSRN: http://ssrn.com/abstract=10182http://ssrn.com/abstract=10182.

[Simmons 2007] International Law and International Relations: An International Organization Reader. By Beth A. Simmons, Richard H. Steinberg Contributor Beth A. Simmons, Richard H. Steinberg Edition: illustrated Published by Cambridge University Press, 2007. ISBN 0521861861, 9780521861861

[Sinha 2006] Geoinformatics: Data to Knowledge By A. Krishna Sinha Contributor A. Krishna Sinha Published by Geological Society of America, 2006 ISBN 0813723973, 9780813723976

[Sörensson 2006] Eén and N. Sörensson. The MiniSat page, http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/, Accessed 2006

[Soros 2008] The New Paradigm for Financial Markets: The Credit Crisis of 2008 and What It Means By George Soros Edition: illustrated Published by publicaffairs, 2008. ISBN 1586486837, 9781586486839

[SOX 2002] Sarbanes-Oxley Act of 2002. Second session. One hundred seventh congress of the United States of America. 23 January 2002.

[Spivey 1992] Spivey, J. M. 1992 The Z Notation: a Reference Manual. Prentice Hall International (UK) Ltd.

[Subirana 2006] Subirana, B. And Bain, M. 2006. Legal programming. Commun. ACM 49, 9 (Sep. 2006), 57-62. DOI= http://doi.acm.org/10.1145/1151030.1151056

[Tarantino 2006] Manager's Guide to Compliance: Sarbanes-Oxley, COSO, ERM, COBIT, IFRS, ASEL II, OMB A-123, ASX 10, OECD Principles, Turnbull Guidance, Best Practices, and Case Studies By Anthony Tarantino Edition: illustrated Published by John Wiley and Sons, 2006 ISBN 0471792578, 9780471792574

[Tarantino 2008] Governance, Risk and Compliance Handbook: Technology, Finance, Environmental, and International Guidance and Best Practices By Anthony Tarantino Contributor Anthony Tarantino Published by John Wiley & Sons, 2008 ISBN 047009589X, 9780470095898.

[Tipton 2006] Information Security Management Handbook By Harold F Tipton, F. Tipton, Micki Krause Edition: 5, illustrated, revised Published by CRC Press, 2006. ISBN 0849395615, 9780849395611

[Valente 1999] Valente, A.: Legal Knowledge Engineering. IOS Press, 1999.

[Waddams 2005] Cases and Materials on Contracts By S. M. Waddams, Waddams, S. M., 1942-, Waddams, Trebilcock, mccamus, Neyers, Waldron Edition: 3 Published by Emond

Montgomery Publication, 2005 ISBN 1552391663, 9781552391662

[Walter 1988] Computer Power and Legal Language: The Use of Computational Linguistics, Artificial Intelligence, and Expert Systems in the Law By Charles Walter, University of Houston Program on Law and Technology Edition: illustrated Published by Quorum Books, 1988 ISBN 0899303064, 9780899303062

[Weiss 2005] Weiss, M. and Amyot, D. 2005. Design and Evolution of e-Business Models. In Proceedings of the Seventh IEEE international Conference on E-Commerce Technology (July 19 - 22, 2005). CEC. IEEE Computer Society, Washington, DC, 462-466. DOI= http://dx.doi.org/10.1109/ICECT.2005.35.

[Wintgens 1998] Legisprudence: A New Theoretical Approach to Legislation : Proceedings of the Fourth Benelux-Scandinavian Symposium on Legal Theory By Luc Wintgens Contributor Luc Wintgens Edition: illustrated Published by Hart Publishing, 1998 ISBN 1841133426, 9781841133423

[Wintgnes 2007] Legislation in Context: Essays in Legisprudence By Luc Wintgens, Philippe Thion Edition: illustrated Published by Ashgate Publishing, Ltd., 2007. ISBN 0754626679, 9780754626671

[Wright 2000] Wright, B. And Winn, J. K. 2000 Law of Electronic Commerce. 4th. Panel Publishers.

[Yavatkar 2000] Yavatkar, R., Pendarakis, D., and Guerin, R. 2000 A Framework for Policy-Based Admission Control. RFC. RFC Editor.

[Yu 1995] Yu, E. S. 1995 Modelling Strategic Relationships for Process Reengineering. Doctoral Thesis. UMI Order Number: AAINN02887., University of Toronto.

[Yu 1997] E. Yu. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97) Jan. 6-8, 1997, Washington D.C., USA. pp. 226-235.

[zChaff 2008] http://www. princeton .edu/~chaff/zChaff.html, Accessed 2008.

[Zelkowitz 2001] Zelkowitz M. Advances in Computers. Academic Press, 2001 ISBN 0120121557, 9780120121557.

[Zhang 2001] Zhang , L., Madigan, C. F., Moskewicz, M. H., and Malik, S. 2001. Efficient conflict driven learning in a boolean satisfiability solver. In Proceedings of the 2001 IEEE/ACM international Conference on Computer-Aided Design (San Jose, California, November 04 - 08, 2001). International Conference on Computer Aided Design. IEEE Press, Piscataway, NJ, 279-285. By Panos Periorellis Edition: illustrated Published by Idea Group Inc (IGI), 2007. ISBN 1599046393, 9781599046396.