Automatic Generation of Goal Models from Regulations

Rouzbahan Rashidi-Tabrizi

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

Master of Electrical and Computer Engineering

Under the auspices of the
Ottawa-Carleton Institute for Electrical and Computer Engineering

uOttawa

University of Ottawa
Ottawa, Ontario, Canada
October 2013

Abstract

Organizations in many domains such as healthcare, finances, telecommunications, education, and software development, must comply with an ever-increasing number of regulations, including laws and policies. In order to measure compliance to regulation, several recent approaches propose modelling regulations using goals and indicators. However, creating goal models for regulations is time consuming and prone to errors, especially as this is usually done manually. This thesis tackles this issue by automating some of the steps for creating goal models, and by offering better ways to create graphical views of goal models than what is currently available nowadays in goal modelling tools.

The notation used in this thesis is the Goal-oriented Requirement Language (GRL), which is part of the User Requirements Notation standard and is supported by the jUCMNav tool. The concepts of regulations and their indicators are captured using a tabular presentation in Comma-Separated Value (CSV) files. An import mechanism is added to jUCMNav to automatically create regulation goal models from such files. The imported GRL model can then by visualized using novel features that enable the addition of multiple views/diagrams in an efficient and usable way.

Acknowledgment

First and foremost, I would like to state my sincere gratitude to my supervisor, Professor Daniel Amyot, for his unconditional continuous support during my studies. It was an honor to have him as a supervisor and to have had a chance to gain experience while working beside him. I truly admire him as a thinker and as a wonderful human being. This work would never have gone anywhere without his guidance, patience, motivation, enthusiasm, support and encouragement. I will never forget the experience of working with him and studying under his supervision, and I will always look up to him.

I would like to state my deepest appreciation to Gunter Mussbacher for being the source of brilliant ideas, and for his generosity in sharing them with me when I needed them the most. I also thank him for guiding and helping me when I was lost or desperate, and most importantly for coaching me while writing several great papers. I will always be thankful for his kindness.

Also, many special thanks to my sweet dear Maryam for helping me, advising me, picking fights with me from time to time, and supporting me while I was studying or working on my thesis. Without her besides me, I would have had a hard time finishing my thesis.

I also appreciate my beloved family for their unconditional support and for being so understanding during my studies. To my father, Darioush, who has always supported me without any boundary, and who taught me how to live; to my mother, Jamileh, who has always loved and guided me throughout my life; and to my evil genius brother, who is always like a hammer in my head, ready to wake me up and bring me up in my darkest hours.

I would like to thank NSERC and industrial partners for funding this work, as well as the University of Ottawa for helping me during my transfer and for providing a great environment for study and research. I would finally like to thank all of my friends, who were supportive and provided me with ideas.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
ACM	Association for Computing Machinery
CSV	Comma-Separated Value
CTFC	Commodity Futures Trading Commission
EMF	Eclipse Modeling Framework
GEF	Graphical Editing Framework
GORE	Goal-Oriented Requirements Engineering
GRL	Goal-oriented Requirement Language
HIPAA	Health Insurance Portability and Accountability Act
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
ITU-T	International Telecommunication Union
KPI	Key Performance Indicator
NFR	Non-Functional Requirement
RSL	Requirements Specification Language
UCM	Use Case Maps
UML	Unified Modeling Language
URL	Unified Resource Locator
URN	User Requirements Notation
XML	eXtensible Markup Language

Chapter 1. Introduction

In this thesis, a novel approach for the automatic generation of goal models from regulation documents is presented. Tool support is developed for this approach, with additional features to more easily create diagrammatic views of a complex goal model.

This introduction chapter discusses the context of this work and research questions. In addition, it gives an overview of the research methodology, thesis contributions, and publications based on this work.

1.1. Context and Motivation

Nowadays, regulation compliance has become extremely important. *Organizations* must comply with a variety of constantly emerging and evolving laws, policies, and other types of *regulations*. Non-compliance may result in important consequences, including financial penalties, loss of reputation, lawsuits, and even imprisonment. Organizations want to be able to *measure* their level of compliance against relevant regulations, in order to determine where to invest their efforts in increasing the compliance level while at the same time meeting other organizational objectives [54][55][56]. At the same time, *regulators* want to measure the level of compliance of the organizations they regulate, often to assess the efficacy of their own regulations [8][61].

Keeping in mind that regulation text is often written in natural language (e.g., plain English), measuring compliance with regulations directly is a time-consuming, error-prone task where the end result is often unreliable and difficult to use. Recent approaches for compliance measurement attempt to solve some of these issues by *modelling* the regulations in a way that highlights the *goals* of regulations [21], and even *indicators* measuring these goals [54]. However, creating goal models with indicators from regulations is still a manual process, which is itself time consuming and prone to errors. For ex-

ample, Shamsaei reported that converting regulations to goal models took her about 20 minutes on average [56].

In this context, this thesis proposes a new approach to automatically generate goal models (including indicators) from regulation text represented in a structured, tabular format. Although other structural presentations of regulations could be used (e.g., XML), a tabular representation has the benefit of being easily editable by office tools such as Microsoft Excel, which are commonly used in industry and regulated/regulating organizations. In addition, further tool-supported capabilities are developed to enable the creation of graphical views of these goal models in a simpler way. This results in regulation models that are simpler and faster to create, more complete, and less subject to errors. These models can then be further used for regulation compliance monitoring and analysis.

1.2. Research Questions

There are three important and specific research questions investigated in this thesis:

1.2.1 To what extent can we automate the generation of goal models from regulations?

The main and most important question is whether the generation of goal models can be automated from regulation text, and to what extent. The goal models are actually regulation models that are meant to reflect regulations found in textual documents. However, considering the fact that the creation of goal models for textual regulations is currently being done manually at high costs, achieving automation (even partially) here can bring high benefits to organizations and to regulators. Note that we are more interested in capturing the *structure* of a multi-regulation document with goal models than the *semantics* of individual regulations. This is in line with the approaches of Shamsaei [56], Ghanavati [24], and Siena [58]. In this context, we are not interested in Natural Language Processing techniques to analyse sentences composing regulation text.

1.2.2 Does this automation lead to models that are faster to produce and of higher quality?

It is desirable to know whether an automated approach can result in simpler and faster generation of goal models from regulation text, and whether the resulting models can be more complete and accurate than manually produced ones. To get a quantitative assessment here, one would need to do a full usability study. However, this thesis limits its assessment to qualitative results and a comparison with existing work, and a usability study is left as future work.

1.2.3 How can the proposed approach help the regulatory compliance process?

Since, the focus of this approach is on a rather important part of the regulation compliance process, it is relevant to understand how the generated goal models can help that process, especially in view of how the process of measuring the compliance of regulations in different domains is facilitated.

1.3. Methodology

The methodology used for conducting this research contains the following six steps, which are in line with Hevner's *Design Science* [28]:

1. Problem selection.
2. Study of the problem in the literature, with a gap analysis.
3. Proposition of a solution to the problem, with implementation.
4. Application of the solution to validation case studies (from different domains).
5. Analysis of the results.
6. Communication of the results.

The thesis starts with an introduction to a challenging issue in regulatory compliance, namely regulation modelling, and how to make this issue simpler to handle.

To analyse what has been done prior to this work in relation to the generation of regulation goal models, a literature review was conducted using an approach inspired from Kitchenham's systematic literature reviews [37]. This review describes the most common issues in regulation modelling and available approaches to address them. The use of goal modelling, and in particular of the User Requirements Notation (URN) [34], was studied in this step.

The automatic generation of regulation goal models from regulation text is proposed as a new solution to answer the common issues and research questions. The approach is developed for URN and implemented as an extension of an existing URN modelling and analysis tool, namely jUCMNav [35].

The tool-supported generation is applied to real regulations from three very different domains, namely banking, healthcare privacy, and aviation security.

The results are analysed in terms of the experience gained with these regulation domains, of a comparison with closely related work against the criteria used for the literature review, and of observed limitations and threats to the validity of the approach.

Finally, the research results were published in four papers [3][5][47][48] and the implementation was made available in the release 5.4.0 of jUCMNav.

1.4. Thesis Contributions

The **major contribution** of this thesis has two parts.

1. The first part is a new approach to generate goal models from regulation text automatically. This part requires the user to structure the regulation text in a tabular format (where the columns represent the main concepts, the rows the individual regulation parts, and the format is a *Comma-Separated Value* file, CSV) according to guidelines. This document can then be imported automatically to the jUCMNav modelling environment as a goal model.
2. The second part is a new approach to import *key performance indicator* (KPI) *conversion groups* into regulation goal models. With this feature, a user can provide (in a separate tabular/CSV representation) groups of mappings between evaluation values of a regulation domain and GRL satisfaction values in a goal model.

The import allows for these mappings to be associated with regulation indicators in goal models. This essentially enables one to provide different types of indicators in a GRL model.

This contribution is important as it enables people who are not experts in GRL/jUCMNav to create regulation models. In a sense, the proposed tabular representation of regulations in a Domain-Specific Language that enables users from organizations and regulatory agencies to use conventional office tools (e.g., Microsoft Excel), with which they are most familiar. Often, regulations are already available in a database and/or a tabular format, which is even a better start point for the user.

A **minor** but useful **contribution** of this thesis is a set of new jUCMNav functionalities for investigating dynamically goal models and creating diagrammatic views along the way. These functionalities complement the import mechanisms nicely as they allow one to create GRL diagrams of a regulation model easily, but they are also generally useful outside the context of regulation modelling.

1.5. Publications

In relation with the last step of the research methodology (communication of the results), this thesis research has led to four publications so far:

1. Badreddin, O., Mussbacher, G., Amyot, D., Behnam, S.A., **Rashidi-Tabrizi, R.**, Braun, E., Alhaj, M., and Richards, G.: "Regulation-Based Dimensional Modeling for Regulatory Intelligence". *6th Int. Workshop on Requirements Engineering and Law (RELAW 2013)*, Rio de Janeiro, Brazil, July 2013. IEEE CS, 1–10.

   This paper presents a *methodology* about creating and using goal models (including the automated approach introduced in this thesis) and business analytics for regulatory intelligence, that is, for measuring the performance of organizations and for evolving regulations where required.

2. **Rashidi-Tabrizi, R.**, Mussbacher, G., and Amyot, D.: "Transforming Regulations into Performance Models in the Context of Reasoning for Outcome-Based Compliance". *6th Int. Workshop on Requirements Engineering and Law (RELAW 2013)*, Rio de Janeiro, Brazil, July 2013. IEEE CS, 34–43.

   This paper gives an overview of the approach proposed in this thesis, with a case study from the banking domain (discussed in Section 7.1).

3. **Rashidi-Tabrizi, R.**, Mussbacher, G., and Amyot, D.: "Legal Requirements Analysis and Modeling with the Measured Compliance Profile for the Goal-oriented Requirement Notation". *6th Int. Workshop on Requirements Engineering and Law (RELAW 2013)*, Rio de Janeiro, Brazil, July 2013. IEEE CS, 53–56.

   This paper applies the approach from this thesis to a benchmark regulation case study related to healthcare privacy in the United States of America (discussed in Section 7.2).

4. Amyot, D., **Rashidi-Tabrizi, R.**, Mussbacher, G., Kealey, J., Tremblay, E., and Horkoff, J.: "Improved GRL Modeling and Analysis with jUCMNav 5". *6th International i* Workshop (iStar 2013)*, Valencia, Spain, June 2013. CEUR-WS, Vol-978, 137–139.

   This paper highlights some of the recent features of jUCMNav, including those for efficiently creating diagrammatic views of models introduced in this thesis.

1.6. Thesis Outline

This thesis has eight additional chapters.

- **Chapter 2,** *Background,* provides general background concepts and information on URN, jUCMNav, and regulatory compliance;
- **Chapter 3,** *Literature Review,* covers previous and related work on the use of goals for regulation modelling;
- **Chapter 4,** *Automatic Import of Regulations,* presents the first part of the main contribution of this thesis, with a focus on regulations;

- **Chapter 5,** *Automatic Import and Creation of KPI Conversions,* presents the second part of the main contribution of this thesis, with a focus on indicators;
- **Chapter 6,** *Manipulation of Goal Diagrams,* provides the second and minor contribution of this work;
- **Chapter 7,** *Case Studies,* includes application of the approach provided in the thesis to three different domains, with an analysis of the threats to validity;
- **Chapter 8,** *Testing and Comparison*, discusses the testing strategy and results for the implemented software, and includes a brief comparison to closely related work;
- **Chapter 9,** *Conclusions,* summarizes the contributions of this work, relates them to the research questions, and provides future work items.

Chapter 2. Background

This chapter presents background concepts and technologies that the reader is required to know to have a better understanding of the other parts of this thesis. In the first section, the User Requirements Notation (URN) is briefly described. In Section 2.2 one of URN sub-languages, namely the Goal-oriented Requirement Language (GRL), is described in more details. Section 2.3 focuses on jUCMNav, a graphical editor for modelling and analyzing URN models. Section 2.4 finally discusses regulations and compliance concepts, often in a modelling context.

2.1. User Requirement Notation

The User Requirement Notation (URN) is an international standard from the International Telecommunication Union (ITU-T). URN is a graphical notation supporting requirements elicitation, modelling, analysis and specification [4][34]. URN can be used in different types of systems, such as reactive and distributed systems. The first version of this standard was released in 2008, with a second version 2012 that contains additional concepts and features. More specifically, URN can now be used to measure the satisfaction level of intentional elements and the performance of business process [56].

URN combines two complementary modelling views: the Goal-oriented Requirement Language (GRL) and Use Case Maps (UCM). The way URN approaches requirements specification is not to answer *how* questions in the way that languages like UML answers. URN tries to answer *why, what* and *who* questions about the system. Focusing on these questions and leaving out other details related to *how* questions, software engineers can work at an abstract level during requirements specification and modelling [1]. GRL is used to model intentional elements such as goals, social elements such as actors, and their relationships. UCM is used mainly to specify causal scenarios of responsibilities and architectures.

2.2. Goal-oriented Requirement Language

The Goal-oriented Requirement Language (GRL) is a graphical modelling language for capturing intentional concepts, non-functional requirements (NFRs) and stakeholders. GRL has roots in other goal-oriented modelling languages [56]. It inherits actors, intentional elements and links concepts from the *i** framework [65] and the concept of evaluation mechanism from the NFR framework [14]. GRL makes connection between requirements and business objectives and it helps identify alternatives, explore potential decisions, and document rationales. Software engineers can combine UCM elements with GRL elements to ensure alignment between goals and scenarios, and analyse goal models using quantitative and qualitative evaluation algorithms [2].

A GRL model is visualized as a graph composed mainly of four different categories of elements:

1. *Intentional elements*, which contains goals, softgoals, resources, beliefs, and tasks. During analysis, intentional elements are given a satisfaction level (qualitative or quantitative), either directly in a strategy definition, or through the execution of a propagation algorithm.
2. *Intentional links*, which contain decomposition, dependency, contribution and correlation relationships. Decomposition itself is categorized into three different categories: AND-, XOR-, and OR-decomposition. Contributions can have a qualitative contribution type, or a quantitative contribution level.
3. *Actors*, which is the mean to present systems and stakeholders.
4. *Indicators* (or *KPIs*), which are a recent extension of GRL used in performance analysis. Indicators convert a real-life value (e.g., in dollars) to a GRL satisfaction value according to a comparison with an expected target or to a mapping function.

These elements are depicted in Figure 1.

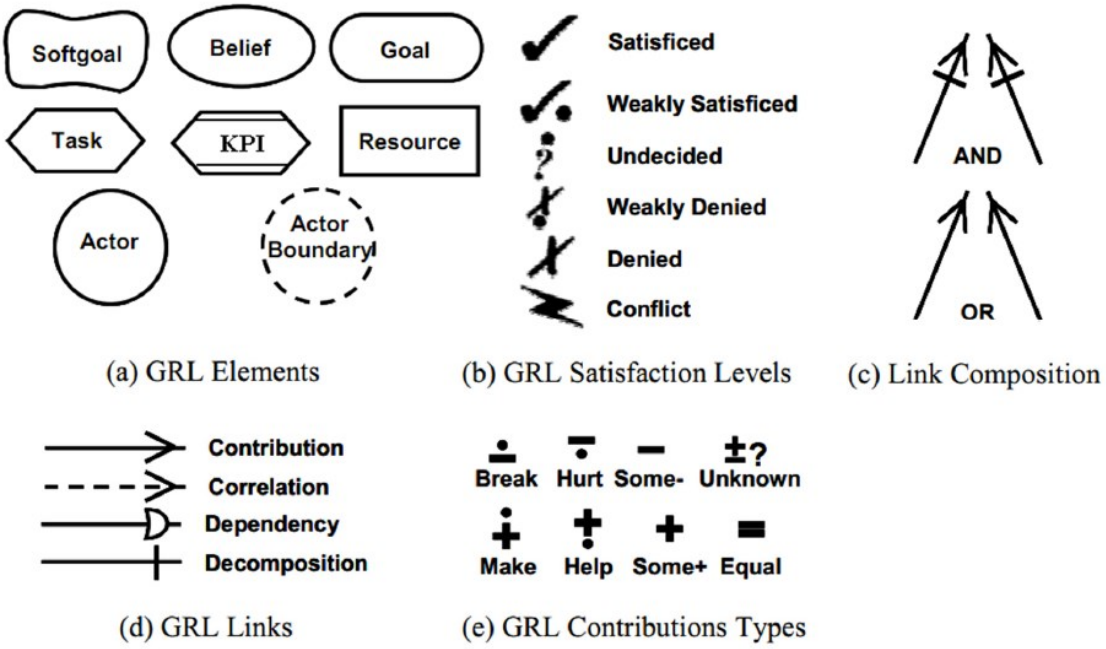


Figure 1 GRL Notation Elements [4]

A *softgoal* is an intentional element that is often used to model NFRs or quality aspects (e.g., security), and cannot be fully achieved. In this matter a softgoal is different from a goal. A softgoal needs to be further decomposed or refined into other achievable and measureable elements. So, by measuring those elements, the softgoal itself can be measured, although *indirectly*. A *goal* is quantifiable, often in a binary way, and can be satisfied fully. Goals are often used to model functional requirements. A *task* is used to model a solution to achieve goal or softgoal. In that sense, tasks are often elements of goal operationalizations. *Resources* are used by a task to achieve and satisfy goals and/or softgoals. *Beliefs* capture explicit rationales and assumptions of stakeholders.

GRL links have different types and can be used to connect intentional elements. *Decomposition* links can be used to decompose an intentional element into sub-elements. The different types of decomposition are AND, OR and XOR. *Contribution* is the other type of GRL links that connects intentional elements to each other. There are two major types of contributions: qualitative and quantitative. A qualitative contribution has a value describing whether the contribution is positive/negative and sufficient/insufficient, with labels such as *make, help, some+, break, hurt* and *some-*. A quantitative contribution has an integer value between −100 and +100 (according to standard URN). A *correlation* is

similar to a contribution but is often used to model side-effects between intentional elements of different actors. A *dependency* link is used to model a dependency between two actors [1][56]. A sample GRL model (graph) is illustrated in Figure 2.

Another important concept in GRL is the *strategy*, which is a specific initialisation of satisfaction values for some of the intentional elements of a GRL model. The satisfaction values can be specified as quantitative or qualitative values. These values are usually annotated with an asterisk (*) in a GRL diagram, and tools may also use dashed contours to highlight the initialized elements and differentiate them from non-initialized elements (see Figure 2). A *GRL evaluation mechanism*, also called *propagation algorithm*, propagates the initial values to other intentional elements in the GRL graph (through the various links) and calculates the satisfaction values of these elements. Color feedback helps visualize whether intentional elements are satisfied (greener) or dissatisfied (redder) at a glance. Moreover, there is another concept in GRL called *Importance*, used in some intentional elements inside actors. Importance values (qualitative or quantitative) are used to calculate the overall satisfaction level of their actors. Different strategies can be defined and compared to determine suitable decisions and trade-offs, or to enable performance analysis. In GRL, there are many different evaluation algorithms that can used, including qualitative, quantitative and hybrid evaluations [2][24][56].

In Figure 2, the use of a hybrid evaluation algorithm is depicted. This sample model captures the structure of a regulation and will be explained later. Note however that it contains eleven goals, eight indicators, and three resources. Decomposition, contribution, and dependency links are used here. Moreover, color coding of the GRL diagram is demonstrated in Figure 2 and shows the satisfaction level of each intentional element.

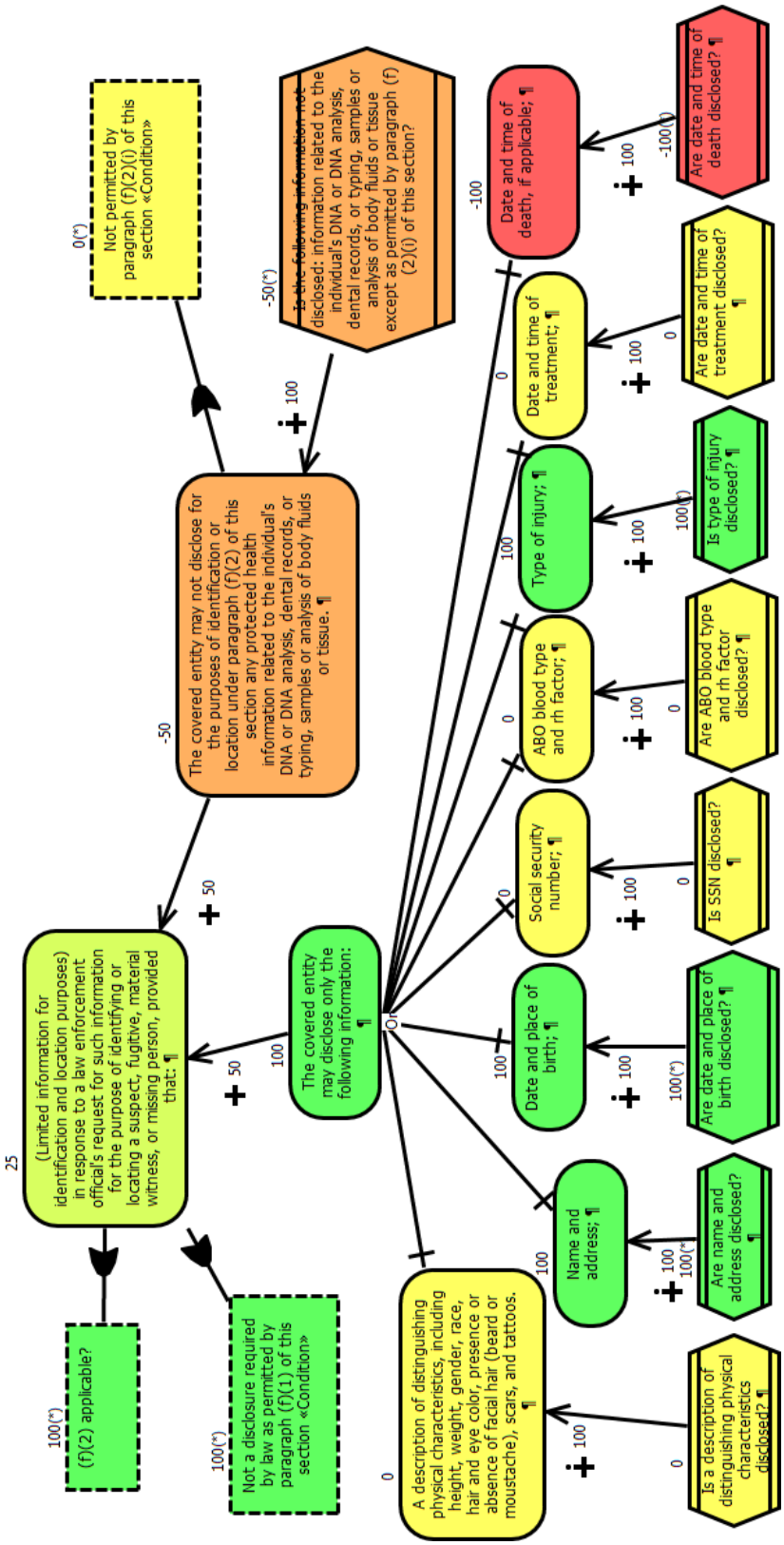


Figure 2 A Sample GRL Model (Graph)

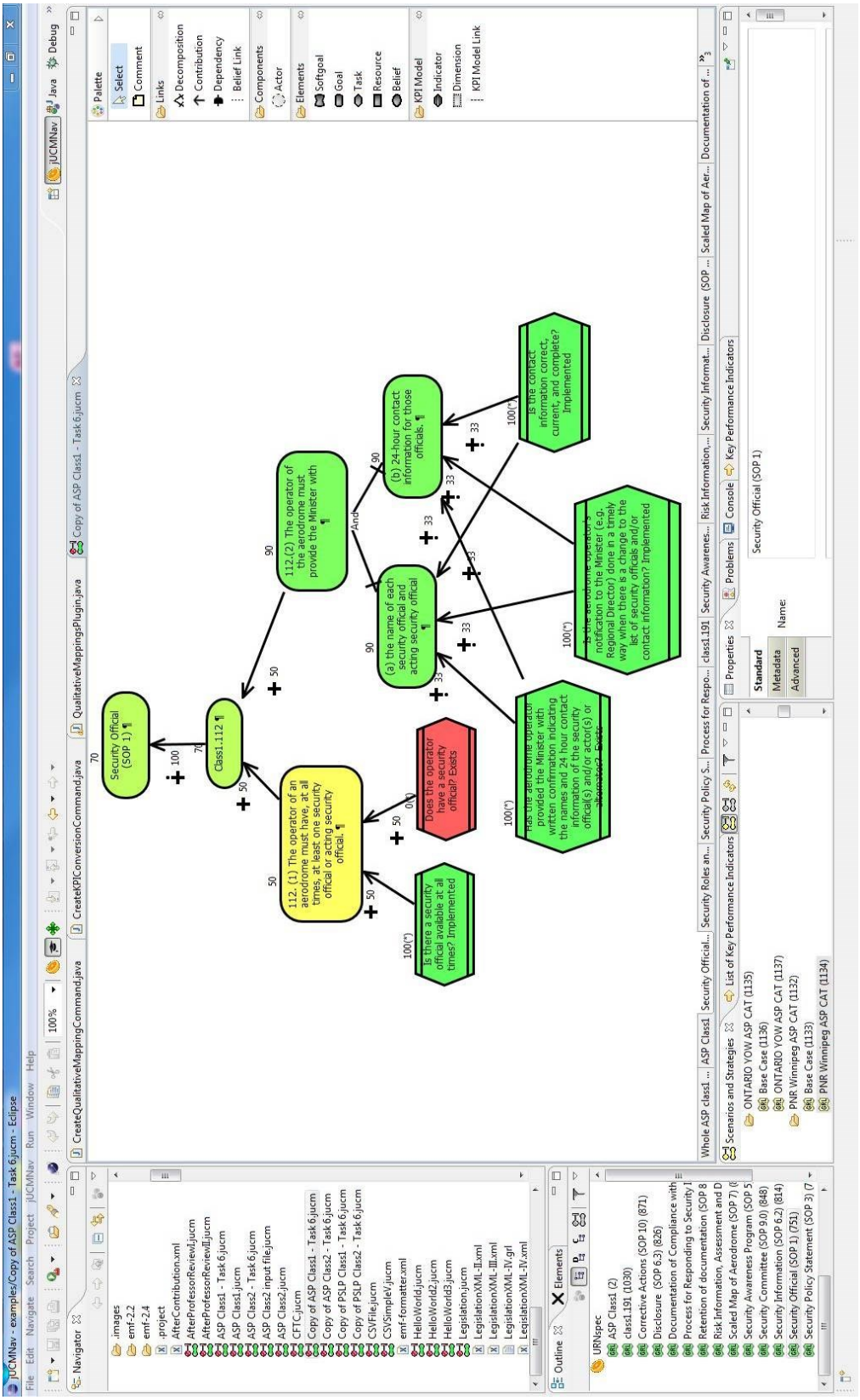


Figure 3 jUCMNav Snapshot

2.3. Tool Support for URN – jUCMNav

jUCMNav is a graphical tool that helps create, analyse, transform, and validate requirement models in URN. jUCMNav is a free, Eclipse-based open source graphical editor is developed with the Eclipse Modeling Framework (EMF) and the Graphical Editing Framework (GEF) [35]. jUCMNav provides support modelling for both components of URN, namely GRL and UCM. Moreover, GRL evaluations and strategies and UCM scenarios can also modelled and executed in the editor. Figure 3 shows a GRL graph modelled with jUCMNav.

2.4. Regulations and Compliance

Regulation modelling and analysis have been studied by many researchers over the past few decades. This section presents *common* approaches and issues in this area, whereas the next chapter will focus more on a systematic comparison of *goal-oriented* approaches.

Kiyavitskaya et al. [39] have observed many challenges related to regulation modelling. They observe that regulations are written and organized in legalese. By this they mean that regulations are vague and ambiguous by nature, in order to better cope with unknown future situations. Consequently, requirements in regulations are hard to capture. In addition, they are often highly spread all around regulation documents. On the other hand, these requirements are necessary for assessing the compliance of processes, software, or any other under application domain. Regulations are also written in *many different natural languages* (e.g., in French and English in Canada) and processing them introduces many challenges for tools. Moreover, the cross-references in regulation documents lead to many *traceability issues*. Finally, because of the prescriptive nature of regulation documents, using text mining and annotations is often necessary for (semi-) automated analysis [39].

According to the work of van Gog and van Engers [62], there are many problems with laws written in natural language with regard to ambiguity, which cause different people to assume multiple meanings from a law. In this context, they suggest that if one

could transform laws into a precise specification, which can be processed by computers, legal compliance could be assessed automatically. In their work, they study the possibility of *transforming* legislation into formal models (specifications). Using their approach, existing ambiguities can be found and fixed before even starting the transformation process. They focus on *traceability* between a model and its originating document. Their transformation is done in two steps. First, they provide a structure for the initially unstructured legislation (law), then they create models for legislation using their parser. However, they used their approach in only one domain, and never discussed its performance.

Breaux and Gordon [11] created an open system to make requirements available to policy makers, business analysts and software developers. They try to describe laws and legislation using a Requirements Specification Language (RSL) to make designing, analyzing, navigating and manipulating easier "using formal structure, patterns and metrics" [11] for all their stakeholders. Also, they are able to visualize requirements relationships across different categories. They take into consideration the fact that legislation exists in "multi-jurisdictional" environments, hence conflicts are a point they have considered in their work. Traceability and evolution of the laws and legislation, as is considered in the work of van Gog and van Engers [62], are the other features they had under consideration. In their system, Breaux and Gordon use limited formalization to make analysis straightforward and possibly more reasonable in terms of computation and undecidability. The analysis is done using Description Logic and graph theory. They take into consideration prioritization of the requirements that apply to the business. Also, cross-references within and among laws is addressed in their work to get rid of repetitions in the RSL system. These relations between and among documents are used to demonstrate the requirements relationships (REFINES, EXCEPT and FOLLOWS [11]) and are implemented by means of different commands in expressions, but contributions and dependencies are absent from these relationships. The authors show that the characteristics of the domain of the laws could also be implemented by defining actors and objects. Their system is fed the plain text of a law and, via tagging, converts it into RSL expressions. They have tool support for their system and are able to generate a model from the initial RSL expressions. The model can be visualized in different formats such as HTML,

XML and GraphML [11]. They have recently extended their work on RSL to include a method to *translate* IT regulations into a legal requirements coverage model used to make coverage assertions about existing or planned IT systems [25].

Rifaut [49] introduced a "performance-based regulatory regime" that can be considered as an outcome-based regulatory framework. He handles issues such as the introduction of new rules or laws, found in documents regarding operational aspects, into the "regulatory regimes". Moreover, he and his colleagues tried to create a measurement framework to be able to measure the compliance of legislation in a system. In this framework, they considered goal-oriented methods and "regulatory conversion" concepts. With regard to conversion, *agreement*, *validity* and *separation of concerns* were the main obstacles that were identified by Rifaut. The concept of *measurement* reflects the outcome of the (main) goals in the models. Rifaut captures the measurement model by studying criteria in the rules and legislations and also by using "business process reviews and interviews" [49]. The measurement model can grow according to the number of business process. One of the interesting things in this approach is that it separates concerns in measurements. Since this approach uses both goal and softgoal concepts, "fulfillment of a goal is measured, whereas the fulfillment of a softgoal is evaluated" [49]. The main issue with this approach is the time required to create all the necessary compliance models, and tool support is absent. Reusing models can help mitigate this issue and avoid continuous model creation. The domain of applicability of this approach does not belong to traditional prescriptive regulations.

Breaux and Baumer [12] discuss that security in customer personal information has become a serious issue and government laws and regulations lead to new sets of requirements for information systems. Compliance to such security requirements turns out to be an important and challenging task. Although this paper highlights many important issues, it does not cover how to do modelling or compliance.

Breaux and Lotrionte [10] mention that laws are continuously changing in response to technologies development and to social and political issues. Moreover, they show that these triggers for updates can be presented in pattern languages like regular expressions. However, the complexity of such pattern languages is a concern they tried to address by aggregating patterns together. In their approach, they can recognize contradic-

tions and conflicts in policies. For this purpose, they use a formal language that captures data flows. To perform goal prioritization, they use both positive and negative norms.

In their work [41], Maxwell and Antón introduce production rule models to obtain knowledge about software requirements from regulatory text in a way such that models are usable and understandable by all stakeholders. They develop a methodology to produce rule models from regulation text as their primary focus is to help software engineers get a better understanding of requirements. They follow previous works that mainly focused on rights, permissions and obligations and make models in way that all these deontic modalities are presented. Their approach introduces some advantages over prior work, including 1) being comprehensive in terms of coverage of the domain, and 2) providing *traceability* [41]. One of the interesting features of their methodology is that it can be implemented iteratively. The main goal of the rule models is to create a new way of communication between different stakeholders by means of "tracing query answers back" [41] to regulation text. They create models from regulation text instead of "modelling legal reasoning". There rule models are presented in Horn clauses with logical operators [7]. All these rules are combined to make a knowledge base and queries are being used to communicate with this knowledge base. The rules are used for requirements compliance checking. For this purpose, as soon as requirements are extracted from regulation text, they can be checked against the rules and compliance can be investigated. Also, evaluation of models can be achieved when "errors and ambiguities are discovered in the model" [41]. They use Sherman's solution [57] with some modifications to implement traceability in their methodology. As a result, evolution of rule models is possible in their work. They also try to translate regulation text to the desirable format and then refactor the rules to reduce the size of rule models by finding repetitive patterns and similarities.

Maxwell and Antón [42] also take a closer look into the rule model production component of their previous work [41]. They believe that, for compliance, software engineers need to figure out natural language interpretation and that requirements must be traceable for this purpose. They claim that there have been many approaches used for developing tools and techniques for legal compliance, but they never provided any guidelines on how to define "natural language compliance requirements from law" [42]. They

introduce a tool-supported framework that generates compliance requirements from regulation text in two different steps: modelling and analysis. By modelling, they try to generate rule models from regulation text, and by analysis, they try to investigate the requirements against the previously-generated rule models. As a result of using their framework, software engineers can easily investigate compliance of requirements, which in turn means having a better overall understanding for all stakeholders. They create production rules in Prolog, since this language is suitable for knowledge presentation. One of the key features of Prolog is backtracking, used here to enable querying for the rule models. An initial query is treated as a top-level goal and then, using backtracking, the value of the goal gets determined. The authors have identified some limitations of this approach where, in some cases, they lose pieces of information or they introduce ambiguity.

Young and Antón discuss that the business practices of organizations have to comply with policy documents [64]. Requirements must be captured from policy documents to make the building of "policy-compliant" software possible. In their work, using commitments, privileges and rights, they derive requirements from policy documents. They work on policy documents that are different from regulation text in some ways. In policy documents, points that an organization must commit to are clearly expressed. These commitments must be made available to users of the organizations. However in regulation text, notions of rights and obligations are used more often. Yong and Antón believe that policy-based documents are more trustable than law-based ones, since the latter do not provide sufficient coverage of requirements. Moreover, they believe policy-based documents describe procedural practices in a better manner [64]. One of the other aspects of their work is a traceability feature, for backward and forward tracking of commitments, rights, and privileges with requirements. Their commitment analysis approach is composed of three different steps: 1) parsing of the input policy document into a set of policy statements, 2) classification into one of the twelve different classes of policies based on statement attributes, and 3) transformation of classified statements into requirements. The last step is done using some requirement templates.

Boer et al. discussed the necessity of having electronic versions of regulation text and legislation [6]. In their work, they try to develop a framework in which they can present legal documents and their structure and content. They define structure and elements

of documents in schemas that enable document validation. However, since there are many different types of documents, different schemas for vocabularies are required. With this framework, Boer et al. claim they achieved incredible improvement in the processing and management of legal documents. They also provide a feature for reasoning by applying metadata in documents. They criticize the lack of language support and of evolvability of current legal documents, and they believe this is caused in part by outdated standards from legal publishers and legislators. They develop "a generic open standard" [6] to improve the maintenance and management of legislation. In their XML schema, they try to develop a universal format to demonstrate the relations between legislations, "search and filtering and version management" [13]. Additionally, using a variety of technologies, this format can be manipulated in a way that can be used in "knowledge presentation, code generation, rule generation and verification of relevant contents" [13]. Moreover, using their framework, schemas can be designed in such way that multilingual regulations are supported. The framework provides some attributes, such as date-publication, date-enacted, etc., that are used to keep track of versions of documents.

2.5. Summary

This chapter reviewed background information useful to understand this thesis. First, URN was introduced, with an emphasis on its goal view (GRL), which is used to model regulations in this thesis. The jUCMNav modelling and analysis environment for URN was briefly presented as it will be extended in this thesis with additional features for regulation modelling. Finally, important literature on regulations, compliance and their relationships, especially from the requirements engineering and modelling areas, was highlighted. It can be observed for the last part that many types of modelling approaches exist, targeting diverse representations (e.g., Prolog, goal models, and XML schemas) with different source documents (e.g., laws, policies, and regulations) and different purposes (e.g., analysis, querying, compliance assessment, and version management), and that the construction of models remains a challenge, especially when some degree of automation is targeted.

The next chapter will focus on a more systematic literature review related to the *generation* of *goal models* from *regulations*, which is the main topic of this thesis.

Chapter 3. Literature Review

In order to gain a deeper understanding of existing work related to automatic *goal model construction from regulations*, this chapter presents the results of a more systematic literature review in that area, with an approach inspired from Kitchenham's [37]. Comparison criteria are first defined in Section 3.1 in order to enable the identification of common weaknesses (i.e., gap analysis). The search engines and queries used are then described in Section 3.2. Relevant approaches are highlighted and compared in Section 3.3, which is followed by a summary.

3.1. Criteria

To get the best out of reviewed research articles, there must be some criteria considered to analyse and sort them out. These criteria are selected based on important factors used to answer the research questions presented in this thesis (see Section 1.2), and to identify potential gaps and opportunities in existing work:

- **Automation**: The main research question of this thesis is concerned with the possibility of automating the generation of goal models from regulation texts. Hence, this factor is used to capture whether such feature exists in reviewed papers.
- **Decomposition**: During the generation of goal models, we need to figure out the relations between regulations in input text. *Decomposition* is one important type of relation. The structure of regulations in textual documents can be used to help infer such relations.
- **Contribution**: This is another type of relation between parts of regulations that must be provided in the construction of goal models, in order to measure how well parts of a regulations are being complied with, at various levels of granularity.

- **Indicator**: In a regulation compliance context, it is important to connect sources of information and test/inspection results to the regulation objectives, and this can be done with indicators.
- **Actor**: In most regulation domains, regulators express which regulation targets which stakeholder/actor. In generated goal models, having regulations bounded to their respective actors will lead to more accurate diagrams.
- **Traceability**: It is important for users to trace regulations in source text to regulation goals in generated goal models (for compliance analysis, understanding, or maintenance), especially as regulations evolve over time.
- **Metadata**: In many cases, regulations text contains important information that can be used to generate better models and enable advanced analysis. To capture this information, metadata attached to goal model elements can play a decisive role.
- **Diagram**: Goal models can enable visual analysis, so whether diagrams are available in existing approaches is a criterion.
- **Evolution**: To have precise goal models from regulation text that changes over time, model evolution needs to be supported.
- **Alternative language**: In many domains, regulations are presented in multiple languages (e.g., in French and English in Canada). Therefore, generated goal models should be presentable in multiple languages.
- **Condition**: Conditions (under which certain regulations apply) exist in most regulation texts, and they must be handled properly in goal models.
- **Class (Stereotype)**: In some domains, different classes of entities will be targeted by slightly different regulations. Some regulations can also target multiple classes. Such regulation classes need to be captured in generated goal models.
- **Tool**: Tools are needed to support the proposed approaches. Often, an approach is proposed without any tool to implement it.
- **Extensibility**: All the approaches need to evolve over time. So, extensibility is a major concern in developing any approach.

- **Modelling language**: Which specific goal-oriented modelling language is used also represents useful comparison information.

These criteria will be used to evaluate, study, and analyse the researched articles in this literature review.

3.2. Review Methodology

In this literature review, three important and trustable search engines were used to find conference and journal papers relevant to goal-oriented regulation modelling:

- IEEE Xplore[1]
- ACM[2]
- SpringerLink[3]

To find related articles with these search engines, query expressions composed of important keywords were used in the *advanced search mode*.

For the purpose of this thesis, we were interested in general goal model generation, and then more specifically in regulation modelling. Thus, after some early attempts, the following five query expressions were considered to be efficient and fruitful enough, and were used on the three search engines:

- "goal model generation"
- "regulation modeling"
- "automatic goal modeling"
- "regulation compliance"
- "regulation models"

[1] http://ieeexplore.ieee.org/Xplore/

[2] http://dl.acm.org/dl.cfm

[3] http://link.springer.com/

"Modelling" (Canadian/UK spelling) was also used instead of "modeling", without substantial additional results. We used those five query expressions in the mentioned search engines. Since each query often returned hundreds or thousands of results, they were sorted by relevance (ranks) and only the first hundred papers were considered based (as many became quickly irrelevant). After a quick read of titles and contexts of the initial 300 papers (in order to quickly filter out irrelevant ones), 279 publications were chosen for further study. Through elimination of duplicates and inspection of the abstracts, introductions and conclusions, most papers were filtered out, with the remaining 19 closely-related groups of papers being summarized in the next section.

3.3. Generation of Goal Models for Regulations

The results are presented in two parts: the first one covers goal model construction in general (outside the regulation modelling domain) whereas the second part focuses on goal modelling of regulations.

3.3.1 Goal Model Generation in General

Several papers discussing goal models and their potential and limitations (in general) raise interesting points related to model construction/generation and usage.

Faily works on the integration of goal models with User-Centred Design [18]. He tries to investigate how User-Centered Design concepts such as *Persona Cases* can be modified in such way that they could represent goal models. He suggests that *i**-like languages could be used to capture design rationale, and his assumption is to have alignment between *i** and persona cases documented. As part of his objectives, Faily wants to investigate the *generation* of goal models automatically from existing analysis. He picked the Goal-oriented Requirement Language (GRL) as his target goal modelling language. He also used a tool named CAIRIS [17], first to achieve the necessary alignment between goal models and personas, and second to generate GRL files for jUCMNav. He manipulated and updated CAIRIS to provide support for associations between personas and use cases and to add elements necessary to goal models and to generate corresponding GRL

files. This work demonstrates the feasibility of generating GRL models from other representations.

Mussbacher et al. [44] have provided an overview of shortcomings of GRL goal models (from the 2008 version of the URN standard) for requirements engineering activities. Graphical goal models are good for understanding and for visual analysis feedback, but they are usually flat and unstructured, and can become quite large and complicated. This complexity causes problems for model creation and maintenance. They proposed that by providing an alternative textual language or a *tabular presentation* with a suitable development environment, this model creation/maintenance issue could be mitigated. Moreover, they brought up an additional problem with contribution values, which are hard to determine when creating goal models (and sometimes *relative weights* are simpler to use). They also mentioned the problem of reflecting *real-life observable values* into satisfaction values in goal models, which led to the formalization of indicators (KPIs) in the 2012 version of URN. Additionally, the highlighted the importance of having elements tagged or carrying *metadata*, grouped, and linked to each other in various ways in order to support the extensibility of goal models. They finally observed that there was insufficient tool support for advanced user features such as *interactive queries* to explore models interactively and extract particular views.

As in the previous paper, Ellis-Braithwaite et al. [16] support the ideas of *measurable goals* (with metrics/indicators) and of better ways of handling contributions (e.g., with relative weights) in languages such as GRL. They even suggest adding confidence intervals to contribution levels. Moreover, they criticized the way the aggregated contributions are added up to feed into parent goals in upper levels of a goal graph. They highlighted the need for meaningful (and user-defined) *scales*, as well as a means to capture when goal satisfaction is *exceeded*.

Strohmaier et al. [60] mention common issues in modelling in *i** for requirements, like *cost of modelling* and *complexity of models*. Also, they identify the usage of *patterns* as an available solution for these issues and also an approach to capture knowledge about requirements. However, they bring up the downsides of using patterns and they set their focus of their work on "exploring challenges related to patterns" [60]. With two case studies, they try to answer their research questions. They conclude that using patterns

helps reducing the modelling effort and model complexity, *if* patterns are already available (which is not really the case in the regulation domain).

Oliveira et al. [45] consider *i** models as models with a lot of details and complexity, which makes them prone to errors. In their paper, they focused on producing quality *i** models and *preventing defects* during modelling. They introduced a diagnosis process that can be applied to *i** models and claimed that managing complexity (and quality) of models hence becomes possible. No automation is provided however.

Lockerbie et al. [40] tried to model different concepts from operations documents with *i** in order to connect operational concepts with formal simulations. Operations documents contain text and pictures that describe processes and technology. They claim that the *i** (manual) modelling of such documents is feasible, but that this can be a very time-consuming and costly exercise. They also found the *evolution* of models fairly complicated to achieve.

Munro et al. [43] investigated decomposition links in *i** goal models in order to better systematize model construction. They mentioned that achievable goals in a domain are usually different from what modellers considered during modelling, especially when expressed through decomposition. The issue of non-familiarity with the domain and issues with the nature of decomposition relationships introduce *differences* between goal models created by different modellers or domain experts. Such differences can hence be minimized through automated construction.

Franch et al. [19] highlighted several deficiencies related to the construction of goal models, including the complexity of models, traceability between elements, and "lexical and syntactical conventions". They argue that a lack of methodological approach for constructing these models is the main reason for such deficiencies. They also argue that existing (and manual) goal model construction approaches usually end up in different models with many points of noise, contradictions and ambiguities, because people have different views and understanding of the domain (this is similar to the observations of Munro et al. [43]). Moreover, issues like modularity, reusability, scalability, management and traceability were claimed to have little support (as also observed by Mussbacher et al. [44]). Thus, they proposed a methodology to construct goal models in *i** that addresses many of these issues with three steps: (a) domain analysis, (b) constructing the social

system and (c) constructing the socio-technical system. From all these steps, only the last one is iterative and supported by software.

Grau et al. [27] focus on the development of new information systems from existing or legacy ones. They discuss that the intermediate steps in the development process usually benefit from modelling techniques, and they mention that using *i** goal modelling is a good approach for constructing, searching and evaluating alternatives for new systems. They proposed a new and comprehensive approach based on *i** called PR*i*M in order to provide the necessary process for developing and reengineering new information systems and for analysing and evaluating alternatives, systematically. They discuss several benefits including that having *i** used in this framework helps investigating and proposing different solutions for organizations in a systematic way. They have provided guidelines for each approach step to make sure the output of one step was correctly prepared for the next one. In the reengineering phase, different intentional elements may be added or removed to achieve improvement in new system They provided tool support for their approach called J-PR*i*M [26], an Eclipse-based tool with a graphical interface. However, this tool does not show *i** models in a conventional graph but in a tree-like hierarchy.

Elahi and Yu [15] tried to model and analyse security trade-offs using an extension of the *i** framework. As security is competing with other aspects of software design, trade-off analysis can be helpful in examining advantages and disadvantages of a specific design in regard to other system and stakeholder goals. In this approach, conceptual models include (a) "design goals and objectives, alternative operationalizations on goals and their impact on goals and objectives" and (b) "actors who seeks alternative solutions" [15]. They proposed a metamodel covering security concepts and mapped it to *i** concepts. They also provided an analysis method to be used by designers to find a balanced solution. However, complexity of models and automation of analysis remain challenging in this approach.

3.3.2 Goal Modelling for Regulations

Kiyavitskaya et al. [39] paid special attention to gathering rights and obligations from regulation text for software developers. They claimed that extracting correct requirements

from regulation text is a challenging task that needs proper tool support. The approach that they proposed consists of three steps: (1) extracting actors, rights, obligations, etc.; (2) constructing a semantic model from the output of the previous step; (3) transforming this semantic model into software requirements. They provided tool support for the first step of their approach using and extending the *Creno framework* [38]. In other words, what is done by Breaux et al. in [9] is presented as an automated tool in [39] to help "analyzing policy documents". They also claimed their tool works well in a considerable number of domains where even different regulations may apply to a system.

In [33], Ishikawa et al. stated the need for clarification about goals that are defined by laws. They brought up the importance of interpretation of the law, since it may have a significant impact on decision making. They also mentioned the importance of managing goal models (represented with the KAOS language [63]). They proposed an approach for legal interpretation management in goal modelling by means of a new metamodel for legal interpretation. They focused on modelling, analyzing and creating goal models. In term of modelling, they emphasized goal refinement (they used the term *concept* instead of goal) and they made a comparison between obtained *concepts* and their "*actual instances"* [33]. In term of analysis, their approach enables exploring instances for their corresponding concepts and investigating gaps between instances and their expected-to-be concepts. They tried to model refinements as criteria for goal models to impose the least changes possible (and hence simplify management).

Ghanavati et al. [23] looked at GRL goal modelling and how it could support the process of regulation compliance. They discussed how dealing with laws could be troublesome and how relevant documents are complex and could be hard to process and handle, especially when multiple laws are considered [24]. They also brought up benefits of using UCM alongside GRL in a compliance process. In her thesis [24], Ghanavati further provided a compliance framework with a set of metamodel-based GRL model construction guidelines that take advantage of Hohfeldian and deontic modalities. However, it does not take advantage of indicators to measure compliance.

Building on her previous work on business process modelling [55] and goal model families [53], Shamsaei [56] proposed an indicator-based framework with metamodel-based guidelines to develop goal models (including actors) from regulations. She paid a

considerable amount of attention to contribution and decomposition links between goals in generated goal diagrams. She also tailored GRL with tags and constraints to provide more precise modelling and analysis. Still, her proposed guidelines are manual and prone to errors.

Rifaut and Ghanavati [50] showed the necessity of modelling regulations into goal models to be applicable in their measurement framework. This work essentially supplements Rifaut's performance approach [49] with the GRL-based framework of Ghanavati [24], although this is less integrated and not as well supported by tools as Shamsaei's approach [56].

Siena et al. [59] claim that laws and regulations cannot be modelled as goals explicitly. They believe that an "abstraction gap" and ontological differences can cause difference between regulations text and goal models, and the modelling would not be accurate enough. They prefer to separate organization goals from legal *norms*, which is different from the approaches of Ghanavati [24] and Shamsaei [56], where the same language (GRL) is used for laws and for organizations. They also mentioned that conventional goal modelling approaches are not good enough and more advanced approaches with more consideration for conditions and alternatives must be developed (this was also acknowledged by Ghanavati and Shamsaei, who both support conditions in their frameworks).

In his thesis, Siena [58] observes that although there is much evidence that goal modelling contributes positively to efficient requirements engineering, there is little evidence that compliance and requirements are at that point yet. Thus, he introduced a framework to address this issue by providing a conceptual methodology for compliance. One of the elements that he introduced in his framework was "a language for modelling requirements and the impact of legal prescriptions on requirements" [58], named *Nòmos,* an *i**-based language used to connect requirements and law models. Few guidelines are available for the construction of such models.

Ingolfo et al. [32] discussed the ever-increasing demand for legal compliance in software requirements. They talked about their Nòmos-based framework, which provides preference and priority options for stakeholders in regulation models. They highlighted the challenges that software engineers have to face when looking for a way to capture compliance with law, and how this gets more complicated by fast-changing laws. Their

main focus is on how to capture stakeholder preferences and priorities, and "variability of compliance alternatives" in goal models. A very interesting point in their work is an automated reasoning mechanism provided in their tool.

Although the above papers cover many approaches closely-related to ours, the interested reader is invited to read the surveys of Ghanavati et al. [21] and Shamsaei et al. [54] for additional and complementary insight in goal-oriented modelling for regulation compliance.

3.4. Summary

The automatic generation of goal models is challenging from many angles. The papers from Section 3.3.1 have highlighted many language-level challenges that this thesis will address, including the use of a tabular (and convivial) representation for the automated generation of goal models (including metadata), the use of relative weights in contributions, the support of indicators with user-defined scales (including ways to capture when a goal is exceeded) during model creation, and the interactive construction of graphical views from goal model queries. Automation is also believed to minimize some of the differences in models that would be created by different people.

Additional points can be observed from the papers on regulation modelling (Section 3.3.2) and Table 1, which summarizes the paper reviewed in this chapter against the criteria discussed in Section 3.1. Most of the approaches have used (*i**, KAOS, GRL) or extended (Nómos) existing requirement engineering modelling languages. However, only about half of them have provided tools for their respective approaches. Moreover, they mainly have no automation for their model construction approaches.

Although most approaches support decomposition and contribution links, in practice, very few support relative weights on contributions. Extensibility and metadata are weakly supported in general, and very few approaches consider the classifications of regulations, alternative natural languages in models, and traceability to the original text.

Table 1 Literature Review Summary

Article	Automation	Decomposition	Contribution	Indicator	Actors	Traceability	Metadata	Diagram	Evolution	Alternative Language	Condition	Class (Stereotype)	Tool	Extensibility	Modelling Language
[18]	Yes	Yes	Yes	No	No	No	No	Yes	No	No	No	No	Yes	No	*i**
[44]	No	No	Yes	No	Yes	No	Yes	Yes	No	Yes	Yes	Yes	+/-	Yes	GRL
[16]	No	Yes	Yes	No	Yes	No	No	Yes	No	No	No	No	Yes	No	GRL
[60]	No	Yes	Yes	No	Yes	No	No	Yes	No	No	No	No	No	No	*i**
[45]	No	Yes	Yes	No	Yes	No	No	Yes	No	No	Yes	No	No	No	*i**
[40]	No	Yes	Yes	No	Yes	Yes	No	Yes	No	No	Yes	No	No	Yes	*i**
[43]	N/A	Yes	N/A	No	Yes	N/A	N/A	Yes	N/A	N/A	N/A	N/A	N/A	N/A	*i**
[19]	+/-	Yes	Yes	No	Yes	No	No	Yes	Yes	No	Yes	No	+/-	Yes	*i**
[27]	Yes	Yes	Yes	No	Yes	Yes	+/-	Yes	+/-	No	Yes	No	Yes	No	*i**
[15]	No	Yes	Yes	No	Yes	No	No	Yes	No	No	No	No	No	No	*i**
[39]	+/-	No	No	No	Yes	+/-	No	No	No	No	No	No	Yes	Yes	N/A
[33]	No	Yes	Yes	No	No	Yes	No	Yes	Yes	No	No	No	No	No	KAOS
[23]	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	GRL
[60]	No	Yes	Yes	No	Yes	No	No	Yes	No	No	No	No	No	No	*i**
[56]	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	GRL
[58]	No	Yes	No	No	Yes	No	No	Yes	No	No	Yes	No	Yes	No	Nòmos
[32]	+/-	Yes	Yes	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	No	Nòmos
[50]	No	Yes	Yes	No	Yes	+/-	No	Yes	No	No	No	No	Yes	No	GRL
[59]	N/A	Yes	Yes	No	Yes	No	No	N/A	N/A	No	Yes	No	N/A	N/A	N/A

Different approaches offer different advantages and disadvantages according to the criteria provided in Section 3.1. Shamsaei's framework [56] offers the most complete solution so far according to our criteria, and is used as a start point for this thesis. Its main weakness is the lack of automation for model construction, and this will be addressed by our improved and tool-supported approach in the next chapter, which takes as input a tabular representation of regulations easily produced by domain experts. Other approaches by

Grau et al. [27] and Ingolfo et al. [32] are also addressing our criteria well (with some automation), and hence we will consider them for further comparison with our own work in Chapter 8.

Chapter 4. Automatic Import of Regulations

In this chapter, a new automation approach to importing regulations as goal models is described, together with details about its input, output and steps. The Goal-oriented Requirement Language (GRL) part of the User Requirement Notation (URN) standard is being used as the target language for our approach. In the following section, an overview of the approach is provided. Section 4.2 then presents the metamodel of the input of the approach, whilst Section 4.3 defines its concrete syntax based on a tabular presentation that can be stored as a Comma-Separated Value (CSV) file. Section 4.4 provides the necessary steps to produce the output and explains them in detail. Finally, Section 4.5 highlights notable features of this thesis' solution.

4.1. Overview

The main objective of the approach is to automate part of the creation of goal models from regulation documents. A secondary objective is to enable the creation of graphical views of the model easily. This approach, as one of the very first steps of legislation compliance, can be widely and easily applied to a many different domains.

Figure 4 gives an overview of the process of our approach from inputs to output. The main input of the approach is a regulation/policy/law document (with regulations, sub-regulations, rules, conditions, etc.). A secondary and optional input is a list of indicators (KPIs) used to measure compliance to specific regulation parts.

The regulation text must be formatted in a tabular representation in a spreadsheet CSV file (arrow #1 in Figure 4). This transformation can be done manually, or this can be produced automatically if a regulations database exists. This representation is essentially a Domain Specific Language (DSL) for which this thesis defines a metamodel (Section 4.2) and a concrete CSV textual syntax (Section 4.3.1). Thus, the CSV file (or legislations document) is an instance of the DSL's metamodel. This legislations document is a

Comma-Separated Value (CSV) file that reflects the architecture of the original regulation text. It also provides other information, which makes precise automatic goal model creation possible. The CSV file can also include the related KPIs, when available (arrow #2). The choice of CSV as a concrete syntax rather than, for example, XML, is critical to enable non-modellers (e.g., compliance officers) to essentially document a regulation goal model, but efficiently using office applications such as Microsoft Excel[4], which they already know. Although XML/CSV files could be partially constructed automatically from existing regulation databases, such databases are not always available (in which case an Excel-based CSV file is easier to construct).

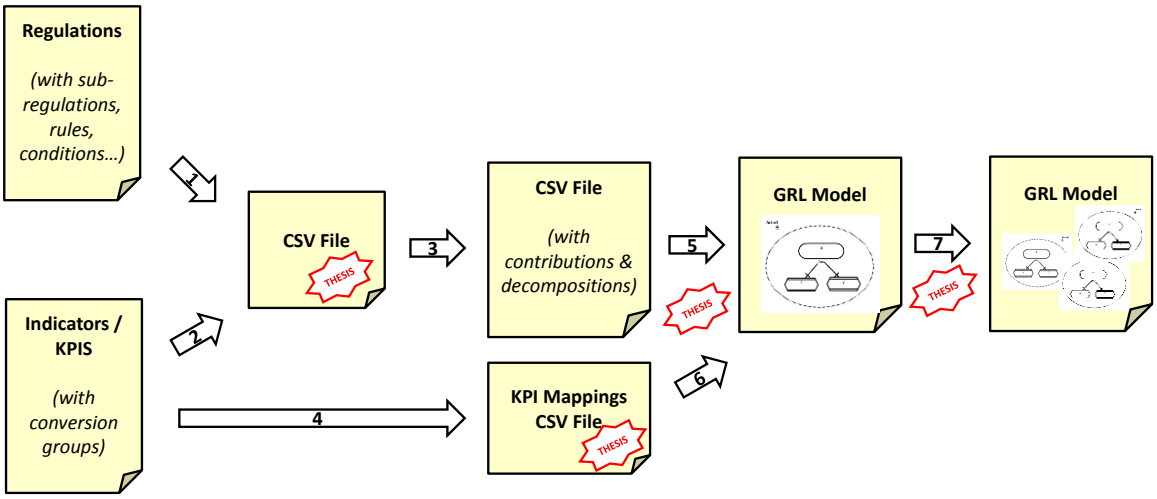


Figure 4 Automatic Goal Model Creation and Manipulation of Goal Models

The CSV file must be augmented manually with information about contribution levels (using relative weights) and non-obvious decomposition relationships (e.g., OR or XOR decompositions not apparent from the structure of the regulation document). This step (arrow #3 in Figure 4) is done manually as it involved parsing and decomposing law texts based on some guidelines (presented in Section 4.3.2) into the proper format. Another manual step involves the creation of a set of mappings for converting KPI types into GRL satisfaction values. Such mappings are stored in a separate and reusable CSV file for

[4] http://office.microsoft.com/en-ca/excel/

convenience (arrow #4). Hence, steps #1 to #4 represent pre-processing work necessary to make sure a natural language regulation document is formatted into the proper tabular presentation (CSV file).

Another part of the main contribution of this thesis is illustrated with arrow #5 and targets the automatic generation of a GRL model (with one global diagram) from the CSV file capturing the regulation text, and from pre-loaded KPI mappings (arrow #6). This is basically a model-to-model transformation. An instance model of the regulation metamodel in the form of a CSV file is transformed into an instance model of URN/jUCMNav. This transformation is covered in this chapter, whereas Chapter 5 is devoted to the particular case of KPI mappings (arrow #6).

The minor contribution of the thesis is shown with arrow #7 and illustrates that many diagrams/views of the regulation GRL model can be created through additional jUCMNav features covered in Chapter 6.

4.2. Tabular Presentation of Regulations

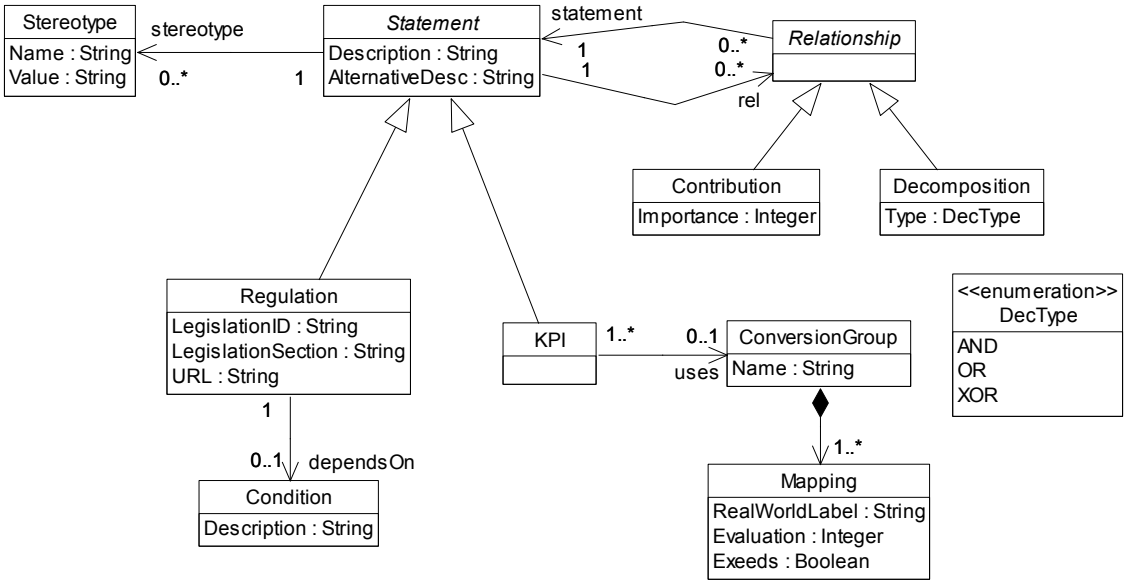


Figure 5 Tabular Presentation Metamodel

The input of the approach is a tabular presentation with a concrete syntax implementable as a CSV file in the Excel environment. The CSV file is basically an instance of the tabular presentation metamodel. This metamodel, inspired from and compatible with Shamsaei's [56], is presented in Figure 5.

Statement is considered as an abstract class that is specialized further to capture *Regulation* and *KPI*. Each Statement has a *Description* and an *AlternativeDesc* (for the statement in a second language, e.g., English and French) as attributes. Regulation has three additional attributes: *LegislationID, LegislationSection* and *URL* (Unified Resource Locator) for traceability to the original regulation text.

Regulations and KPIs can be linked to other Regulations and KPIs by means of *Relationship*s. There are two types of Relationships: *Decomposition* and *Contribution*. A Contribution has an Importance attribute whereas Decomposition has a *Type* (*AND, OR* or *XOR*) attribute. Moreover, Regulations and KPIs may have as many *Stereotype*s as needed. Regulations may also have a *Condition* assigned to them. Finally, as KPIs have their own way of being evaluated, a KPI may have a named *ConversionGroup* that contains at least one *Mapping*. A Mapping contains three attributes, *RealWorldLabel, Evaluation* (on the GRL satisfaction scale) and the Boolean *Exceeds* (set to true when the RealWorldLabel means that we exceed full satisfaction).

All these classes, attributes, and associations have a concrete presentation in one of the two CSV file formats. The data fields used in the regulation CSV format (see arrow #3 in Figure 4) are presented and discussed in detail in Section 4.3. Table 2 summarizes the existing mappings between these data fields and the metamodel's classes and attributes.

An instance of this metamodel in the concrete syntax (visualized with Excel) will be used to illustrate these fields.

Table 2 Mapping between Metamodel Concepts and their Concrete Syntax

Class	Attribute	Data Field in CSV file
Regulation	Description	ENGLISH_DESCRIPTION
Regulation	LegislationID	LEGISLATION_ID
Regulation	LegislationSection	LEGISLATION_SECTION
Regulation	URL	URL
KPI	Description	KPI
Contribution	Importance	IMPORTANCE
Decomposition	Type	DECOMPOSITION
Regulation	AlternativeDesc	ALTERNATIVE_DESCRIPTION
KPI	AlternativeDesc	ALTERNATIVE_KPI
ConversionGroup	Name	KPI_STEREOTYPES
Stereotype	Value	STEREOTYPE and KPI_STEREOTYPES
Condition	Description	KPI_IMPORTANCE

4.3. Creating a Comma-Separated Value (CSV) File

Following the analysis of the regulation text, each and every regulation must be considered to be placed properly in a Comma-Separated Value (CSV) file. In this process, shown with arrows #1, #2, and #3 in Figure 4, a regulation may be decomposed into multiple sub-regulations to reflect the tree-like structure of the regulation text in the tabular representation.

The rest of this section focuses on describing the structure of the CSV file format (the concrete syntax of our metamodel). Parsing (modelling) of regulation text into GRL goal models manually is explained in detail in Shamsaei's guidelines [56].

4.3.1 Comma-Separated Value (CSV) File Structure

There is a specific structure for regulation CSV files that must be followed by the person who parses regulation text. The structure contains some mandatory fields and some optional ones. The optional fields bring considerable amount of flexibility to our approach.

This flexibility makes the creation of goal models feasible, even with a lack of information (e.g., about indicators), especially in the first iterations.

The provided flexibility shows how evolution of the goal models can be achieved by means of iterations. In very first iterations, goal models can be generated with CSV files that carry mandatory fields. In the following iterations, as more information becomes available, optional fields can be filled in the CSV file and more detailed goal models can then be generated. Note that there is no need to iterate if all of the information is available right from the beginning.

A typical regulation CSV file contains the following mandatory fields, ordered from the leftmost column to the rightmost one:

- LEGISLATION_ID
- LEGISLATION_SECTION
- ENGLISH_DESCRIPTION
- ALTERNATIVE_DESCRIPTION
- URL
- IMPORTANCE
- DECOMPOSITION

The regulation CSV file may also have the following optional fields:

- STEREOTYPE
- KPI
- KPI_IMPORTANCE
- ALTERNATIVE_KPI
- KPI_STEREOTYPES

For mandatory fields, some table cells can be left empty without introducing failures, as default values will be used during the import. Optional fields can be entirely left out of the CSV file (i.e., their columns are absent). Some of the optional fields come with additional constraints related to other fields. These issues are described for each field in the followings descriptions. Also, a typical CSV file is illustrated in Figure 18.

LEGISLATION_ID

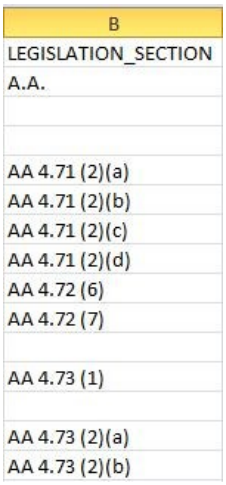
In this field, the identifier (ID) of the legislation/regulation is placed. This ID is a unique number that distinguishes regulations from each other. In some cases where a regulation is decomposed into multiple sub- regulation (e.g., X must do A or B or C), the main and common part of the regulation possesses the unique ID and its sub-regulations (A, B, C) do not require any ID in the CSV file.

	A
1	LEGISLATION_ID
2	90000
3	
4	
5	402318
6	402319
7	402320
8	402321
9	402341
10	402342

Figure 6 Example of LEGISLATION_ID Field in a CSV File

LEGISLATION_SECTION

This field captures the regulation/legislation section identifier. It is used in our implementation to capture the structure of legislation documents, and this structure will be reflected in the generated goal model. This field is mandatory and cannot be left empty for any regulation. Note that the way section identifiers are presented in documents highly depends on the nature and domain of the regulation text. However, they are always hierarchical, and hence a common prefix (e.g., *AA 4.7(2)* in the following example figure) indicates a parent-child decomposition relationship with sub-regulations (i.e., *AA 4.7(2)(a)* and *AA 4.7(2)(b)* are *siblings* at the same depth in the decomposition tree).

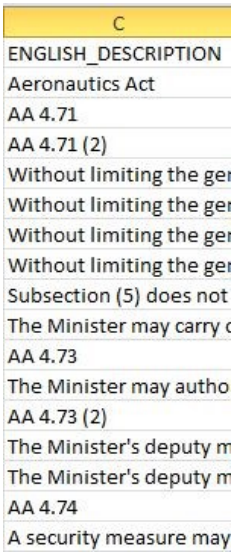


B
LEGISLATION_SECTION
A.A.
AA 4.71 (2)(a)
AA 4.71 (2)(b)
AA 4.71 (2)(c)
AA 4.71 (2)(d)
AA 4.72 (6)
AA 4.72 (7)
AA 4.73 (1)
AA 4.73 (2)(a)
AA 4.73 (2)(b)

Figure 7 Example of LEGISLATION_SECTION Field in a CSV File

ENGLISH_DESCRIPTION

In this field, the description (or name) of the regulation is placed. This field literally contains regulation statements. Hence, it cannot be left empty. Note that although the name of this field is ENGLISH_DESCRIPTION, the content is not limited to English statements.



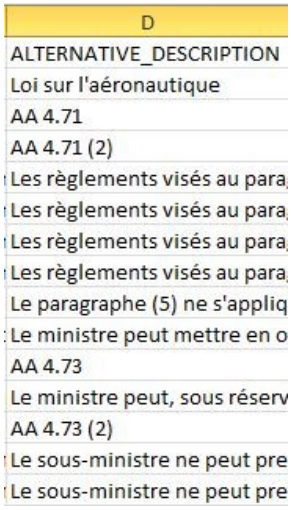
C
ENGLISH_DESCRIPTION
Aeronautics Act
AA 4.71
AA 4.71 (2)
Without limiting the ge
Without limiting the ge
Without limiting the ge
Without limiting the ge
Subsection (5) does not
The Minister may carry
AA 4.73
The Minister may autho
AA 4.73 (2)
The Minister's deputy m
The Minister's deputy m
AA 4.74
A security measure may

Figure 8 Example of ENGLISH_DESCRIPTION Field in a CSV File

ALTERNATIVE_DESCRIPTION

In this field, the same regulation description can be written in a second language. The models hence support two languages (e.g., English French, German, Chinese, etc.), which

is useful in some contexts. For example, in Canada, many laws are available in both English and French. This field can be left empty for regulations, but must be present in the CSV file.



D
ALTERNATIVE_DESCRIPTION
Loi sur l'aéronautique
AA 4.71
AA 4.71 (2)
Les règlements visés au para
Les règlements visés au para
Les règlements visés au para
Les règlements visés au para
Le paragraphe (5) ne s'appliq
Le ministre peut mettre en o
AA 4.73
Le ministre peut, sous réserv
AA 4.73 (2)
Le sous-ministre ne peut pre
Le sous-ministre ne peut pre

Figure 9 Example of ALTERNATIVE_LANGUAGE Field in a CSV File

URL

This field can contain a hyperlink to the original regulation statement. This field can be left empty if no URL is available when regulation text is being prepared.

E
URL
http://www.tc.gc.ca/e
http://www.tc.gc.ca/e
http://www.tc.gc.ca/e
http://www.tc.gc.ca/e
http://www.tc.gc.ca/e
http://www.tc.gc.ca/e
http://www.tc.gc.ca/e
http://www.tc.gc.ca/e

Figure 10 Example of URL Field in a CSV File

IMPORTANCE

In this field, the *relative* value of contribution of the regulation statement to its parent regulation is defined. This value usually is determined by experts of the domain and

shows the *importance* of one regulation over its siblings. For example, if one regulation is three times more important that its only sibling regulation, then *3;* and *1;* would be used, respectively (or 6 and 2, or 9 and 3, etc., as what is important is the ratios). Moreover, a range of values for a contribution also can be defined in this field. This feature usually gets useful when the domain expert has not yet decided on the value of a specific contribution to a parent regulation. To define a range for a contribution, the two extremes must be separated by a *";"*. This field is mandatory, but can be left empty since not all regulations have contributions (e.g., the top-level regulation of a regulation tree).

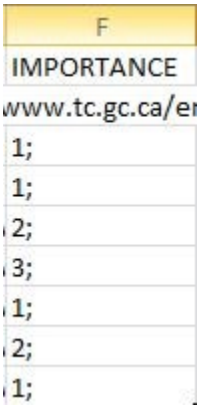


Figure 11 Example of IMPORTANCE Field in a CSV File

DECOMPOSITION

In this field, the decomposition type (And, Or, Xor) of a child regulation is defined. This field is usually defined by the person who parses regulation text. We have not attempted to use natural language processing on the text to extract this information automatically. This is mandatory field, which means that it must be presented in the CSV file. However, it may be left empty, since some regulations may have contributions rather than decomposition relationships to their parent. Note that a regulation cannot have both a decomposition and a contribution at the same time.

Figure 12 Example of Decomposition Field in a CSV File

STEREOTYPE

Regulation statements can be tagged with one or more stereotypes based on the needs of the domain, and this field is used to capture them. This field is not mandatory, so it can be left empty. However, if this column is not needed at all, it must not be placed in the CSV file. A regulation can have multiple stereotypes separated by a *";"*.

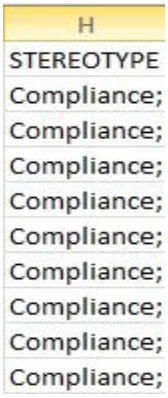


Figure 13 Example of STEREOTYPE Field in a CSV File

KPI

In this field, KPIs used to measure are listed. This field is optional and it can be safely removed from the document if not used at all. However, the presence of this field means that another field, named KPI_IMPORTANCE, must also exist in the CSV file.

A regulation may have multiple KPIs separated by a *";"*. At the end of the last KPI, a *";"* must be placed as well.

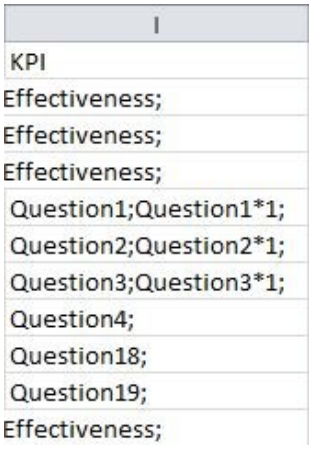


Figure 14 Example of KPI Field in a CSV File

KPI_IMPORTANCE

This field is similar to the IMPORTANCE field, but with two differences. First, this KPI_IMPORTANCE value is a relative value for contribution between KPIs and their parent regulation. Second, a KPI_IMPORTANCE value cannot have any range value defined for it (no range value is considered for KPI contributions). Since a regulation may have multiple KPIs, their KPI_IMPORTANCE must all be defined in one place as well. Hence, these values (whose number must be the same as the number of KPIs) must be separated by a *";"*, with a *";"* at the end.

In this field, *conditions* can be also defined. Sometimes, a regulation statement may have a condition or an exception associated with it. Such conditions are captured in this field. There must be a *";"* placed between the last KPI importance value and the condition. Also, conditions must be prefixed with *"#C_"* and must end with *";"*.

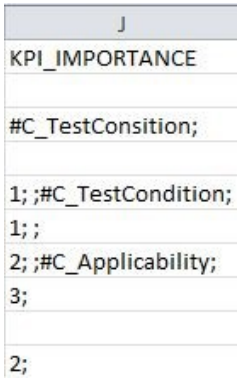


Figure 15 Example of KPI_IMPORTANCE Field in a CSV File

ALTERNATIVE_KPI

In this field, an alternative name or description for a KPI can be provided, if any. Since a regulation may have multiple KPIs, the value of this field may carry multiple alternative names or descriptions, one for each KPI. Thus, these values must be separated by *";"*, with another *";"* at the end.

K
ALTERNATIVE_KPI
question subsidiaire1;qu
question subsidiaire2;qu
question subsidiaire3;qu
question subsidiaire4;
question subsidiaire18;
question subsidiaire19;

Figure 16 Example of ALTERNATIVE_KPI Field in a CSV File

KPI_STEREOTYPES

In this field, defined in the first column but in a row defined after the last regulation statement, stereotypes that qualify a KPI are defined. If there is no stereotype defined for any KPI, this field must not be removed from CSV file. In this field (which actually spans multiple columns), a KPI's stereotypes are placed one after another in different cells of the row, after the KPI description specified in the first column of the row.

KPI_STEREOTYPES			
Question1	Compliance	Effectiveness	#test1
Question2	Compliance	Effectiveness	#test1
Question3	Compliance	Effectiveness	#test2
Question12	Compliance	Effectiveness	#test2
Question19	Compliance	Effectiveness	#test3
Question18	Compliance	Effectiveness	#test3
Question16	#test1		

Figure 17 Example of KPI_STEREOTYPE Field in a CSV File

	A	B	C	D	E	F	G	H	I	J	K
1	LEGISLATION_ID	LEGISLATION_SECTION	ENGLISH_DESCRIPTION	ALTERNATIVE_DESCRIP	URL	IMPORTANCE	DECOMPOSITION	STEREOTYPE	KPI	KPI_IMPORTANCE	ALTERNATIVE_KPI
2	90000	A.A.	Aeronautics Act	Loi sur l'aéronautique	http://www.tc.gc.ca/eng/civilaviation/re			Compliance;Effectiveness;			
3			AA 4.71	AA 4.71		1;		Compliance;Effectiveness;		#C_TestConsition;	
4			AA 4.71 (2)	AA 4.71 (2)		1;		Compliance;Effectiveness;			
5	402318	AA 4.71 (2)(a)	Without limiting the ge	Les règlements visés au	http://www.tc.gc.ca/e	And		Compliance;	Question1;Question1*1;	1;;#C_TestCondition;	question subsidiair
6	402319	AA 4.71 (2)(b)	Without limiting the ge	Les règlements visés au	http://www.tc.gc.ca/e	And		Compliance;	Question2;Question2*1;	1;;	question subsidiair
7	402320	AA 4.71 (2)(c)	Without limiting the ge	Les règlements visés au	http://www.tc.gc.ca/e	And		Compliance;	Question3;Question3*1;	2;#C_Applicability;	question subsidiair
8	402321	AA 4.71 (2)(d)	Without limiting the ge	Les règlements visés au	http://www.tc.gc.ca/e	And		Compliance;	Question4;	3;	question subsidiair
9	402341	AA 4.72 (6)	Subsection (5) does not	Le paragraphe (5) ne s'	http://v1;			Compliance;	Question18;		question subsidiair
10	402342	AA 4.72 (7)	The Minister may carry	Le ministre peut mettre	http://v1;			Compliance;	Question19;	2;	question subsidiair
11			AA 4.73	AA 4.73		1;		Compliance;Effectiveness;			
12	402343	AA 4.73 (1)	The Minister may autho	Le ministre peut, sous r	http://v1;			Compliance;Effectiveness;			
13			AA 4.73 (2)	AA 4.73 (2)		1;		Compliance;Effectiveness;			
14	402344	AA 4.73 (2)(a)	The Minister's deputy m	Le sous-ministre ne peu	http://www.tc.gc.ca/e	And		Compliance;Effectiveness;			
15	402345	AA 4.73 (2)(b)	The Minister's deputy m	Le sous-ministre ne peu	http://www.tc.gc.ca/e	And		Compliance;Effectiveness;			
16			AA 4.74	AA 4.74		1;		Compliance;Effectiveness;			
17	402348	AA 4.74 (1)	A security measure may	Les mesures de sûreté	http://v1;			Compliance;Effectiveness;			
18	402349	AA 4.74 (2)	If there is a conflict betv	Les dispositions des ma	http://v1;			Compliance;Effectiveness;			
19	402350	AA 4.75	For the purposes of prot	Pour la protection du p	http://v1;			Compliance;Effectiveness;			
20			AA 4.76	AA 4.76		1;		Compliance;Effectiveness;			
21	402351	AA 4.76 (a)	If the Minister is of the o	S'il estime qu'il existe u	http://www.tc.gc.ca/e	And		Compliance;Effectiveness;			
22	402352	AA 4.76 (b)	If the Minister is of the o	S'il estime qu'il existe u	http://www.tc.gc.ca/e	And		Compliance;Effectiveness;			
23	402353	AA 4.76 (c)	If the Minister is of the o	S'il estime qu'il existe u	http://www.tc.gc.ca/e	And		Compliance;Effectiveness;			
24	402354	AA 4.77	The Minister may autho	Le ministre peut, sous r	http://v1;			Compliance;Effectiveness;			
25	402355	AA 4.771	An emergency direction	La directive d'urgence e	http://v1;			Compliance;Effectiveness;			
26	KPI_STEREOTYPES										
27	Question1	Compliance	Effectiveness	#test1							
28	Question2	Compliance	Effectiveness	#test1							
29	Question3	Compliance	Effectiveness	#test2							
30	Question12	Compliance	Effectiveness	#test2							
31	Question19	Compliance	Effectiveness	#test3							
32	Question18	Compliance	Effectiveness	#test3							

Figure 18 Sample Regulation CSV File

Additionally, if there is any *KPI Conversion Group* available for a KPI, the name of the group can be added after the KPI description and its stereotypes (if any). The name of the conversion group must be prefixed by '#'. More information about KPI conversion groups can be found in Chapter 5.

A CSV file with all the introduced fields is illustrated in Figure 18.

4.3.2 Parsing Regulation Text

This section provides guidelines for manually parsing regulation text into a tabular representation in a CSV file. These guidelines are based on experience gained with the various case studies used for validating this thesis. They are comprehensive with respect to these case studies, but there might still be unforeseen issues that have not been considered and that will need some innovation from the modeller. The guidelines are as follows:

1. The most important part of the regulation document, i.e., regulation statements and headers, must be placed under the ENGLISH_DESCRIPTION field. If there is any alternative language description, it must be placed immediately after (on the row) in the ALTERNATIVE_DESCRIPTION field.
2. A regulation statement always has a section ID indicating to which section it belongs. This value must be placed in the LEGISLATION_SECTION field. However, sometimes, regulation statements may have internal IDs as well (e.g., generated from a database). This is in addition to section IDs. Such internal IDs must be placed in the LEGISLATION_ID field. Regulation headers (specified in ENGLISH_DESCRIPTION) do not need section IDs. Section IDs must use a structure where a common prefix represents the parent statement. Thus, random values cannot be considered for section IDs.
3. Based on the textual description of statements in regulation text, their relationships can be determined. This requires the manual parsing of text, and some expertise. Keywords like "*and"* and "*or"* are hints for decompositions. Figuring out relative contribution levels is harder (and usually requires a committee), but usually when the relationship between regulations is not a decomposition, it is a contribution. The value of a default contribution must be set to "1" if it is not defined by domain experts at the beginning of the process. In modelling iterations, this

value can be defined and changed. These values are placed for each regulation in IMPORTANCE and DECOMPOSITION fields (but both must not be defined for one same regulation).

4. The URL (Unified Resource Locator) value for each regulation can be set in the URL field, if such value is available.
5. The STEREOTYPE field is used to show different classes/categories of regulations, and can be exploited in families of goal models [53][56]. Classes are being used by regulators to differentiate different families or sets of regulations from each other. These classes may be mentioned explicitly in the regulation text or may be provided as additional information alongside legislation.
6. Indicators or KPIs may be provided alongside legislations. Therefore, they can be placed directly in KPI field. In some cases, KPIs need to be recognized from legislations text. In this case, it is more challenging and should be done in cooperation with domain experts. Moreover, a translation of KPI descriptions in any other language can be stored in the ALTERNATIVE_KPI field.
7. The contribution value of a KPI to a regulation statement must be placed in the KPI_IMPORTANCE field. Also, if there are any stereotypes available for a KPI, they must be placed in the KPI_STEREOTYPES field. This information usually should be provided by domain experts.
8. There may be a conditions attached to a regulation. They are mostly recognizable with the word like *if, except, etc*. In some cases they are being expressed implicitly, which makes it hard for modeller to recognize and put them in CSV file. Conditions are supposed to be placed in KPI_IMPORTANCE field with proper format.

Only some of the information fields in the CSV file are mandatory. Further, not all the mandatory fields need to be filled. This feature helps the modeller create a CSV file and generate regulation goal models even with little available information about the domain. This also supports the iterative development of goal models as information is acquired.

4.4. Generating a GRL Model

Following the creation of a regulation CSV file, a GRL model can be generated automatically (arrow 5 in Figure 4). This transformation is implemented as an import filter in the jUCMNav modelling tool. The import works in two phases. In the first phase, a CSV file is read and saved into an appropriate data structure, and then some calculations are done to process information and produce the necessary information for creating regulation goal model. The end result of this phase is an XML file with a specific structure that is saved in the same logical path as the CSV file. This file is called a *GRL catalog*, a simple GRL import format initially developed by Roy [52] and supported by jUCMNav [35]. In the second phase, the created GRL catalog file is imported into jUCMNav by its built-in (but now enhanced) filter in such way that a new URN model is created, including one GRL graph containing all the regulations and their connecting links.

The rest of this section gives an overview of the import mechanism and then its two phases are discussed in detail. Many of the data structures and Java methods discussed here are available in Appendix A.

4.4.1 Description of the Import Mechanism

Figure 19 shows the steps of our import mechanism and also tasks associate with each step. The input of this approach is a legislation CSV document, which contains formatted regulations. In Step I of Figure 19, this information is read into relevant data structures using *OpenCSV* [30], and various lists are processed in Step II. In Step III, the structure of the document is inferred, while other necessary calculations related to links between regulations and their KPIs are being done in Step IV in order to make the generation of the GRL graph possible. After these four preparation steps, in Step V (in Figure 19), information elements, including elements and links definitions, of the target GRL catalog file are prepared. This GRL catalog file is then created by storing its structure and information elements as an XML file with the proper extension (*.grl*) in Step VI. Eventually, the generated GRL catalog gets imported into jUCMNav in Step VII, and the final result is a goal graph that reflects the legislation document and its structure. Some of these steps are explained in more details in following.

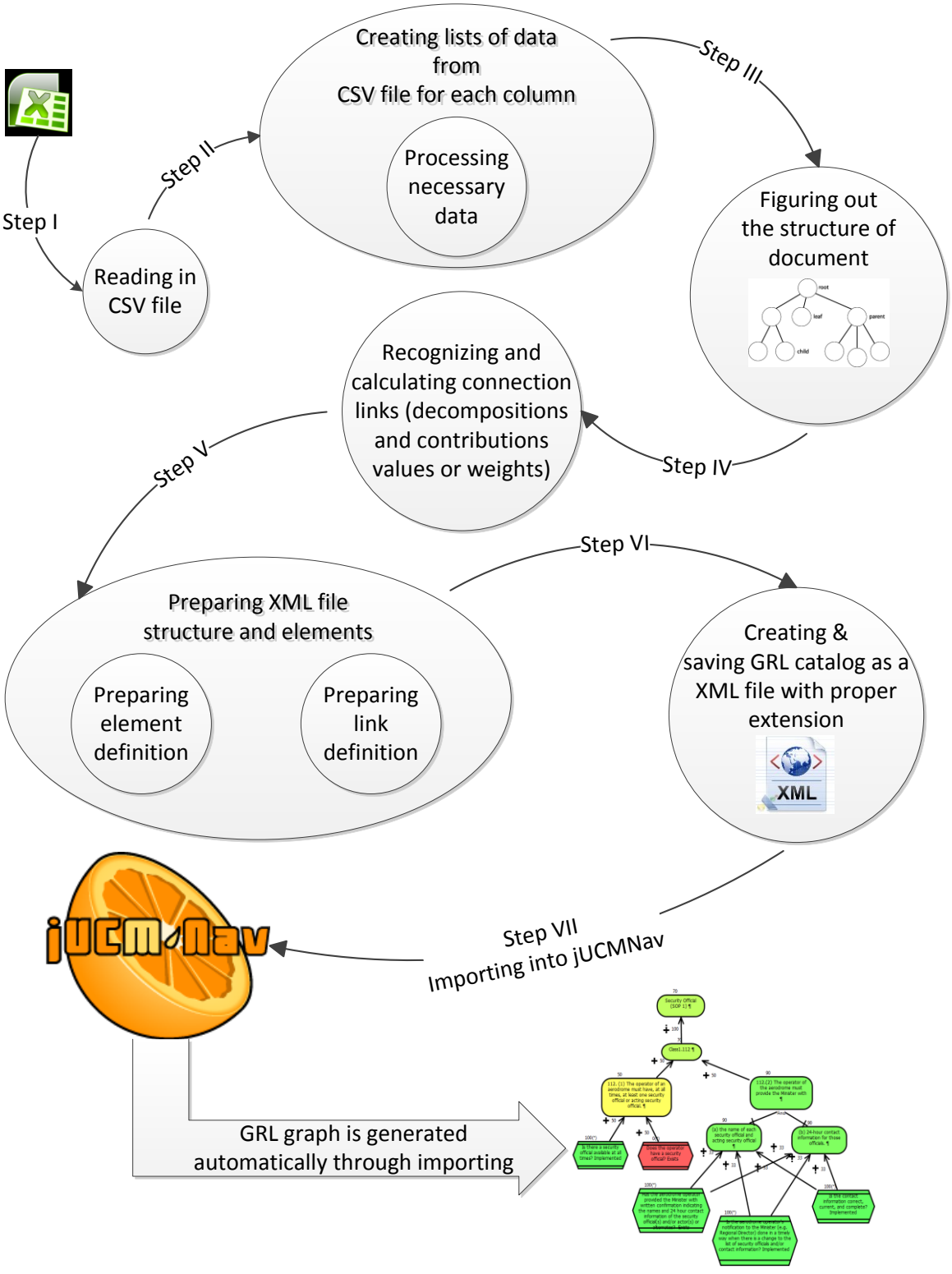


Figure 19 Steps of Import Mechanism

To store the information provided in the input CSV file, different types of lists are used. One very important such list is the one that keeps *Legislation_Section* information. This list is mainly used in Step III, in which a method is called to recognize the structure of document using it. This method, presented in Appendix A.5, puts the structure of the legislations document in a new list and stores it for next steps.

Following the recognition of structure of the document, the connection links between nodes (regulations and their KPIs) get evaluated in Step IV. For this purpose, the information provided in the *Importance* and *Decomposition* fields of CSV file is used. In terms of decompositions, no information processing is necessary. In terms of contributions, the relative weight of each link is provided in the *Importance* field and the GRL value of each link is calculated based on it. So, a method (*calculateContributionValueList*, presented in Appendix A.8) is provided in the implementation to normalize these weights on a 0 to 100 scale and save them into a new list. In this method, relative values entering a node are converted based on the fact that the sum of GRL contribution values of all the entering links to any node in the legislation structure must be 100, as per Shamsaei's guidelines [55]. For example, if there are three incoming contributions with respective relative weights of 3, 1, and 1, then the corresponding GRL contributions become 60, 20, and 20 (summing up to 100). The same process is used for contribution links between KPIs (Indicators) and their related legislation nodes.

In the next step (Step V), the information elements (tags) needed to create GRL catalog (XML) file is prepared and saved into new lists. This step has two sub-steps. In the first one, a list of elements definitions for the catalog file is generated from the processed information. These elements are actually the nodes of the GRL graph. So, considering the nature of a GRL catalog, these elements can be any type of *Intentional Elements* including *Goal, Softgoal, Indicator, Condition and Task.* However, in our domain, we only need the concepts of Goals, Indicators and Conditions as intentional elements. Regulation statements are mapped to goals and inspection questions (KPIs) are converted to GRL KPIs (Indicators). In the second sub-step, a list of link definitions for the catalog file is generated and stored to be used subsequently. These link definitions cover three different types of connection links in GRL: *Contribution, Decomposition* and *Dependency* links (the latter being used to connect conditions to their goals).

In Step VI, these lists are used to create a XML file with a proper GRL catalog structure of information elements. This XML file is stored with the expected extension (*.grl*) to represent a GRL catalog. Finally, this stored catalog is imported into jUCMNav using its built-in mechanism to generate regulations goal models automatically in Step VII. Note that this import mechanism was extended to support concepts metadata, which were not available before.

4.4.2 Reading and Saving the CSV File

An Eclipse plugin was developed and implemented to offer an interface that provides an option for users to import a regulation CSV file and create a corresponding GRL goal model. This plugin extends one of the extension points (*import*) of jUCMNav. This plugin simply reads the information from the CSV file, processes it properly, and provides the result to another part of jUCMNav.

The new plugin takes advantage of OpenCSV [30], a free library with an API that provides convenient functions to read, write, and manipulate data from cells in CSV files. Some of these functions are used to read cells into an appropriate data structure. The initial data structure is a list of strings in which each cell of the CSV file is placed, as briefly illustrated in Appendix A.1.

An instance of the *CSVFile* class is created and it uses a library function to read the CSV file, whose name is provided by the plugin interface. The *ReadFile* function code is shown in Appendix A.2. Also, a cleaning procedure (*deleteBlanks*) is initiated in *CSVFile* to remove all the extra "\n" characters from all the strings in the list. Moreover, a variable named *xmlFileSimple* is introduced that takes care of all the calculations needed to finalize the first phase of goal model creation.

After successful reading of a CSV file into the list of strings, the process of analyzing and manipulating data is started. For this purpose, a new class (*XMLFileSimple*) was added to keep all the information in one coherent unit. This class in the main and most important class of this plugin and it carries a long list of other lists and many useful functions, highlighted in Appendix A.3. Some lists contain the information from the CSV file explicitly, like Legislation_ID, Legislation_Section, IntentionalElement, etc. Other lists are more complex. For instance, *MultipleKpi* is a list of lists that keeps the KPIs as-

signed to a regulation statement. Similar lists include *GoalStereoType* and *KpiContributionValueList*. There is other category of lists that keeps the processed information, for example, to create the XML file of the GRL.

XMLFileSimple also provides many utility functions that are used heavily to process the raw information and generate the XML file. Some of these functions are being used to read and update the information. These are mostly *setter* and *getter* methods, some of which are found in Appendix A.4. Moreover, there is a function that is responsible for creating the catalog XML file and save it in the same directory as the CSV file, with the same name.

One of the most interesting methods in this implementation is *makeRelationList*, shown in Appendix A.5, which is responsible for recognizing the relation tree between the regulation statements. To achieve this, it mainly uses the *LegislationSection* attribute. By end of this method, all the relations between regulations (decomposition, contribution, and condition dependency) are recognized. This method creates and provides a list named *RelationList,* in which regulations and their father/parent node are kept.

4.4.3 Creating a GRL Model from a GRL Catalog

To create a GRL model, we take advantage of a pre-defined facility in jUCMNav [52]. This facility enables a simple import of a GRL catalog file into jUCMNav, and takes care of the creation of a GRL model, with one diagram. Hence, we only need to create a GRL catalog file from the information we have stored into multiple lists from previous steps of this import mechanism.

A GRL catalog is a XML file with a specific structure that is saved with *grl* extension. Some specific fields (tags) must be defined in a GRL catalog to make jUCMNav import it: *element-def, link-def, actor-def* and *actor-IE-link-def.* Figure 20 illustrates these fields in a sample GRL catalog file. In our domain, we do not make use of actors and hence we simply use *element-def* (for intentional elements and indicators) and *link-def* (for contributions, decompositions, and dependencies).

In our plugin, the element definitions for goals (regulations/legislations), indicators (KPIs) and conditions are stored in separated lists. There are also two different lists containing link definitions between goals and between goals, KPIs and conditions.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <grl-catalog author="damyot" description="dghmdghmdghm" catalog-name="ModèleDeJFetAlex">
  - <element-def>
      <intentional-element description="" decompositiontype="And" type="Goal" name="Regulation" id="11"/>
      <intentional-element description="" decompositiontype="And" type="Goal" name="Rule1" id="13"/>
      <intentional-element description="" decompositiontype="And" type="Goal" name="Rule2" id="15"/>
      <intentional-element description="" decompositiontype="And" type="Indicator" name="KPI1" id="19"/>
      <intentional-element description="" decompositiontype="And" type="Indicator" name="KPI2" id="21"/>
      <intentional-element description="" decompositiontype="And" type="Indicator" name="KPI3" id="23"/>
      <intentional-element description="" decompositiontype="And" type="Indicator" name="KPI4" id="25"/>
    </element-def>
  - <link-def>
      <contribution description="" name="Contribution29" correlation="false" quantitativeContribution="25" contributiontype="Help" destid="13" srcid="19"/>
      <contribution description="" name="Contribution30" correlation="false" quantitativeContribution="75" contributiontype="SomePositive" destid="13" srcid="21"/>
      <contribution description="" name="Contribution31" correlation="false" quantitativeContribution="50" contributiontype="SomePositive" destid="15" srcid="23"/>
      <contribution description="" name="Contribution32" correlation="false" quantitativeContribution="50" contributiontype="SomePositive" destid="15" srcid="25"/>
      <contribution description="" name="Contribution33" correlation="false" quantitativeContribution="75" contributiontype="SomePositive" destid="11" srcid="15"/>
      <contribution description="" name="Contribution34" correlation="false" quantitativeContribution="25" contributiontype="Help" destid="11" srcid="13"/>
    </link-def>
    <actor-def> </actor-def>
    <actor-IE-link-def> </actor-IE-link-def>
  </grl-catalog>
```

Figure 20 Sample GRL Catalog

These link definitions can be contributions or decompositions between goals, contributions between goals and KPIs, or dependencies between goals and conditions. Additional lists handle metadata for regulations and KPIs. These lists are simply traversed and output between the XML tags expected by the catalog format. Note that this catalog import plugin was enhanced to support metadata along the way, as this was not supported in Roy's initial version [52].

In the last step, the generated GRL catalog is imported into a new URN model in jUCMNav, with all the regulations as goals, KPIs, conditions, their metadata information, and their connecting links.

4.5. Notable Features of this Approach

In our approach and its implementation, some interesting and handy features were developed, including support for importing models with little or detailed information.

4.5.1 Optional Fields

There are many optional fields provided in our approach. Optional fields are further divided into two categories.

Some fields that are optional must be included (as a column) into the CSV file. For instance, *LEGISLATION_ID* is one such field. Some of its cells can be left empty.

Some other fields (columns) can be removed altogether from the CSV file. For instance, *KPI* is one such field.

4.5.2 Evolution

In our approach, we make the evolution of goal models possible. To be more precise, we can have a very simple, basic and preliminary goal model at the very first iteration of generation from regulations. This is possible only because of the existence of optional fields and cells. In each of the next iterations, small piece of information can be added (e.g., KPIs, metadata, conditions, alternative language…) or existing document can be modified or manipulated to have more precision (e.g., precise relative weights). However,

once a GRL model is regenerated, any graphical/layout manipulation of the model is lost; one should invest in dealing with layout issues and the generation of views only on a stable model.

4.5.3 Tolerance

In our approach, some degree of tolerance is provided. In practice, it was observed that people tend to make mistakes while parsing regulation text and creating CSV files. Some of the mistakes could bring down the whole system. However, we tried to design and implement our approach in a way that reduces the probability of happening of this kind of fatal mistakes. For instance, the recognition of the structure of the model is solely dependent on the *LEGISLATION_SECTION* field of the CSV file. Thus, users cannot introduce inconsistent structures. However, mixing *IMPORTANCE* and *DECOMPOSITION* values would cause trouble in the creation of GRL model.

Providing wrong or inaccurate numbers for relative weights for links between goals and between goals and KPIs will not cause any break down. The relative value might be omitted too, but this issue will not cause any failure either (a default value of 1 will be used). However, the generated model will reflect all inaccuracies in the input.

The original GRL catalog import expects all goal names to be unique. However, providing similar name for regulations and their alternative names in *ENGLISH_DESCRIPTION* and *ALTERNATIVE_DESCRIPTION* will not cause any failure, since a mechanism was developed to detect similar names and make them unique in a simple way (by using the value of the legislation section as a suffix). Other sources of failure could include providing duplicate KPIs. This sort of failure is also avoided by checking for duplicates before proceeding further. The legislations ID is not used in the procedures, so any mistake or duplication in such IDs would not cause any failure.

4.5.4 Applicability

Considering the ease with which CSV files can be created from regulation text, this approach can be easily applied to different regulation domains, as will be shown in the case studies. It could even be used outside of the regulation domain altogether, as long as the

input models have a tree-like shape (e.g., connected goals without cycles). We also believe the approach and its implementation to be easily extensible for domains with unforeseen requirements.

4.6. Summary

In this chapter, the first part of the main contribution of this thesis was described. A domain-specific language for regulation modelling, composed of a metamodel and of a CSV concrete syntax was developed, together with an automatic import mechanism for jUCMNav that produces a GRL model. Users can use familiar tools such as Excel to produce importable representations of regulations. Some of this information can even be generated from regulation database when available. What really requires manual intervention is the selection and provision of weighted relative contribution values, and the parsing of the natural language text for discovering non-obvious decomposition relationships (arrow #3 in Figure 4).

The CSV input format allows for optional fields and empty cells in various contexts, which enable users to incrementally develop or evolve their regulation GRL model. The format and the import filter are also tolerant to various kinds of mistakes.

As the purpose of such GRL models is to enable compliance analysis based on values collected from inspections, tests, and other sources of information, which is typically done through KPIs, we need to specify how collected values are converted to GRL satisfaction values. We hence need to define conversion scales/mappings, and these can differ from KPI to KPI and from model to model (arrow #4 in Figure 4). Before this thesis, jUCMNav did not have a mechanism to import KPI conversion mappings. This concern is addressed in the next chapter, with a CSV representation of these mappings and a second import mechanism corresponding to arrow #6 in Figure 4.

Chapter 5. Automatic Import and Creation of KPI Conversions

This chapter describes the second part of the main contribution of this thesis: the automatic import and creation of KPI conversion groups in the GRL model. Again, a CSV concrete syntax is develop for this part, and jUCMNav is used as the target platform.

5.1. Overview

Once a regulation goal model is created, it needs to be evaluated for enabling compliance analysis. As was observed in Section 2.3, to be able to measure compliance we need to provide necessary inputs from observable data to the goal model. Such inputs are placed in KPIs, whose purpose is to convert real-life values (in hours, dollars, number of events, etc.) into a satisfaction value on the [0...100] GRL scale. However, KPIs may not all use the same conversion function. Conversion mappings hence need to be defined and associated to each KPI. This is very cumbersome and time-consuming to do with the current jUCMNav GUI. The goal of this part of thesis is to enable users to define conversions and associate them to KPIs quickly and in a reusable way.

In the metamodel of Figure 5, each Indicator (KPI) can make use of a named *KPI Conversion Group*. These groups contain *Mapping*s from *RealWorldLabel*s to GRL *Evaluation*s and *Exceed* values. As in the previous chapter, we offer here a simple tabular presentation for this information, which can be manipulated in CSV files. Such files will then be imported in jUCMNav to define qualitative conversion mappings *prior to* importing the rest of the regulation models (with regulations, KPIs, and links), as shown by arrow #6 in Figure 4.

5.2. Format of the KPI Mappings CSV File

The format of the CSV file for KPI mappings contains three different fields named after the *Mapping* attributes of the metamodel of Figure 5: *RealWorldLabel, Evaluation* and *Exceeds*. Furthermore, we need to put conversions into their respective groups.

RealWorldLabel

This field presents the label that is used to express an evaluation value or answer in the domain. A mapping in a given domain can have any number of labels for its evaluations (e.g., Excellent, Good, Average, Bad, Very Bad). Labels do not need to be words; they can easily be numbers or anything (any string) that suits the domain and its evaluation.

To name each of the KPI conversion groups, users need to put the name exactly after this label (*RealWorldlabel*), with the character '-' used to separate the names of the conversions from the label. See Figure 21 for an example with three conversion groups, the first one with 5 labels and the last one with only 2.

	A	B	C
1	RealWorldLabel-First KPI Conversion	Evaluation	exceeds
2	0	0	n
3	1	20	n
4	2	40	n
5	3	100	n
6	4	100	y
7	RealWorldLabel-Second KPI Conversion	Evaluation	exceeds
8	0	0	n
9	1	10	n
10	2	30	n
11	3	80	n
12	4	100	y
13	RealWorldLabel-Third KPI Conversion	Evaluation	exceeds
14	0	0	n
15	1	100	y

Figure 21 Sample KPI Conversion CSV File

Evaluation

This value is the actual GRL satisfaction value for the corresponding *RealWorldLabel* in the goal model. jUCMNav uses this conversion at analysis time. As shown in Figure 21, many labels can have the same evaluation.

Exceeds

This field is a Boolean value (y/n) used in GRL to indicate that the evaluation exceeds the evaluation value representing full satisfaction (100). Such information is useful to describe situations where the evaluation greatly exceeds expectations. In the first conversion group of Figure 21, labels 3 and 4 both indicate full satisfaction (100) in their evaluation, but their Exceeds field differ (4 specifies that expectations are exceeded).

Regulation KPI mappings and their containing groups can hence be defined in a CSV file. However, this information still needs to be mapped to GRL models. A small piece of the URN metamodel related to *KPIConversion*, *Evaluation* and *Strategy*, is illustrated in Figure 22. It can be observed that *QualitativeMappings* (the mappings targeted here) is a kind of *KPIConversion* and that it can contain as many *QualitativeMappings* as needed in the URN model. Each *EvaluationStrategy* can have as many *Evaluation*(s). These evaluation(s) may have *KPIConversion* for their *KPIEvalValueSet*.

In our implementation, if we have mappings available, we import them and assign *KPIConversion* to an existing URN instance, at first, and then we assign them to their respective KPIs in the regulations goal model while importing the legislation document.

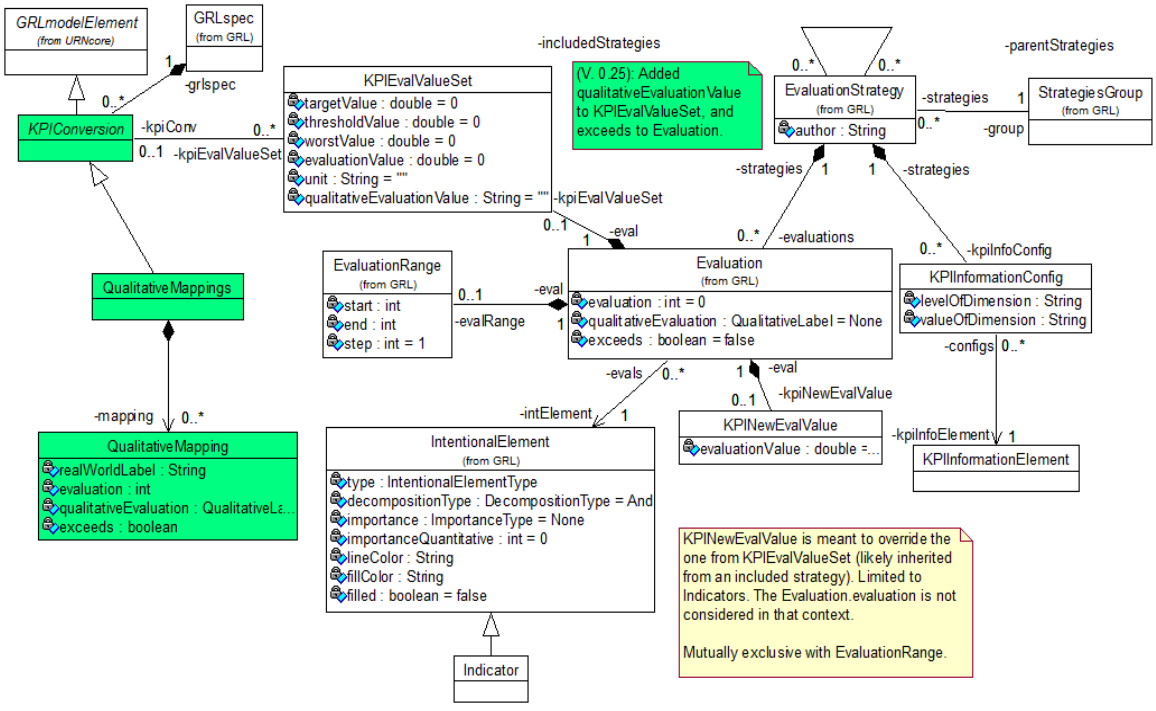


Figure 22 URN Metamodel Extract Related to KPI Conversions [36]

A simple CSV file for KPI conversion groups is shown in Figure 21, and there is another in-use example provided in Table 3. The three *ConversionGroups* defined are used for creating GRL *QualitativeMapping*s. Regarding the URN metamodel in Figure 22, the *qualitativeEvaluation* attribute in *QualitativeMapping* is left empty and the import filter does not provide any value for it. This field will be filled automatically by jUCMNav with appropriate values (an internal conversion from the quantitative *evaluation* value). The *'y'* and *'n'* characters in the *Exceeds* column indicate whether exceeding happens or not, and are converted to *true* and *false* values for the GRL *exceeds* attribute of *QualitativeMapping*.

5.3. KPI Conversion Import Mechanism

The mechanism for importing KPI Conversions is quite similar to the mechanism used for importing regulations into jUCMNav, with a small difference. One of the extension points in jUCMNav was extended and developed like for the previous import filter. However, in this import, *a new instance* of a URN model is first created, and then the process of importing mappings begins. In this plugin, all the information is read into different lists as shown in Appendix A.6. Following a successful reading of the information, the latter is processed into proper lists to facilitate manipulation. The code in Appendix A.6 shows how this information is processed. Each regulation KPI conversion group is converted to a GRL *QualitativeMappings* group in the model, and each mapping is converted to a GRL *QualitativeMapping*.

One issue with the implementation of this part of the thesis was that we had to interact with the Standard Widget Toolkit library of Eclipse for synchronization. Synchronized threads are required in this part of the plugin to achieve the desirable result. We needed to first put a lock on the import process, and then add the conversion groups. After adding the conversion groups and exiting the locked section, the modeling environment recognizes the added groups and shows them in their respective views. This part of code is illustrated in Appendix A.7.

Figure 23 shows a successful import of KPI Conversions in jUCMNav using this mechanism. Note that such conversions/mappings are artefacts that can be reused across regulation models.

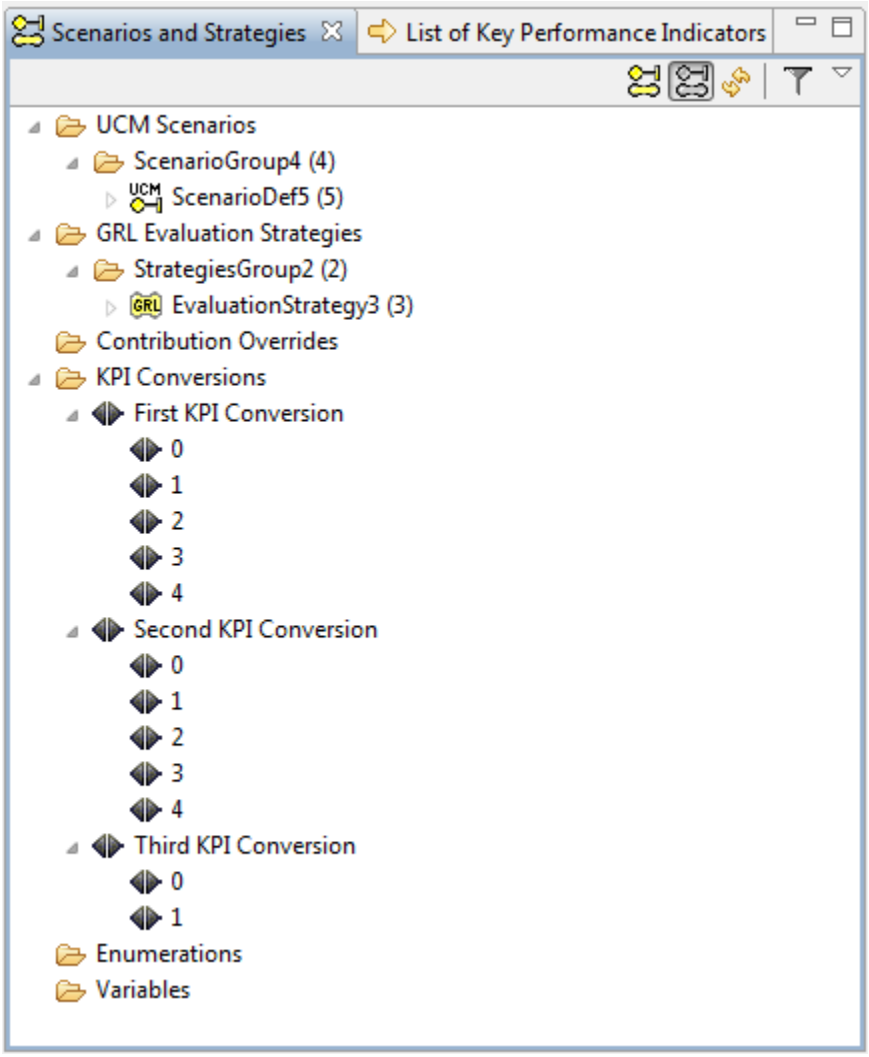


Figure 23 Sample KPI Conversion Import Result

Once KPI Conversions are successfully imported to a new instance of URN in jUCMNav, the regulation CSV file is imported into jUCMNav and the regulations goal model is created. Along the way, we need to assign conversion groups to their respective KPIs automatically. Users need to make sure that the names of KPI conversion groups are provided in their regulation CSV files, and that they match the names that are provided in the KPI mappings CSV file (see Figure 17 for an example).

If there is no KPI mappings CSV file, the regulation document is imported directly, but without support for analysis.

5.4. Summary

In this chapter we explained how KPI conversions can be imported automatically into a URN/GRL model in jUCMNav. This import filter helps users to create desired KPI conversion(s) in a CSV file and then import them in an automated way. This approach is easy to use, much faster than using jUCMNav's GUI, automatic, and less prone to errors. Users can also manipulate conversions details directly from the CSV file. Conversions are easily reusable across models, and conversion libraries could be developed in the future.

After the import of KPI conversions and of regulations, the user is faced with a GRL model that contains only one, potentially very large and complex diagram. Before this thesis, creating sub-views of such model with jUCMNav was a time-consuming and error-prone task, especially in a context where models can evolve. Better features in jUCMNav are needed to enable users to manipulate regulation GRL diagrams and make them more readable and useable. These features are explained in the next chapter.

Chapter 6. Manipulation of Goal Diagrams

In this chapter, a minor thesis contribution is provided. jUCMNav is extended with new features related to the interactive creation and manipulation of diagrams/views of a GRL model. These are generic features, usable for any GRL model and not just regulation models. The next section provides an overview of these features while the other sections describe them in more detail.

6.1. Overview

We propose three new categories of features for jUCMNav to facilitate the manipulation of diagrams in goal models. These features were motivated by the complexity of the GRL models generated from regulations based on the approach of the previous chapters. One large GRL diagram (for example, the one in Figure 24 or the one in Figure 55) is often too complex, and smaller *views* (extracts of the larger diagram) are needed to support understanding and discussions involving human stakeholders. Creating such views using existing *copy and paste* or *drag and drop* features of jUCMNav requires unreasonable time and effort.

Hence, new capabilities are needed to efficiently create diagram views (often involving related intentional elements and actors) of a GRL model. Their requirements are as follows:

- jUCMNav shall support expanding goal diagrams by showing linked elements of a selected intentional element.
- jUCMNav shall support showing the intentional elements of a selected actor.
- jUCMNav shall support showing the containing actor of a selected intentional.

The following three sections cover these three capabilities.

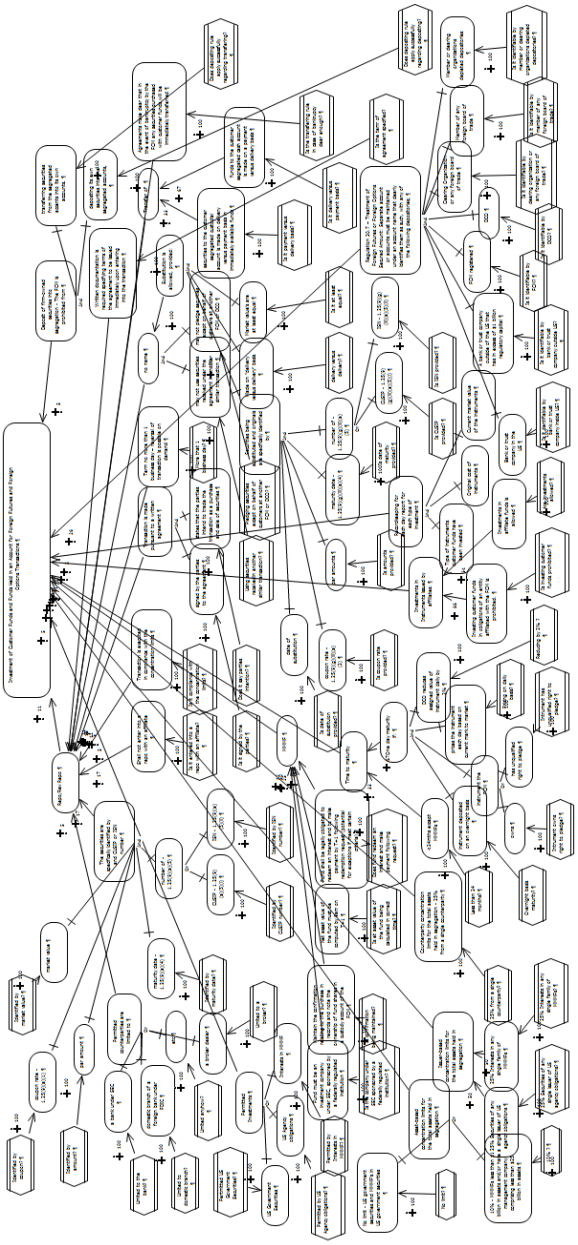


Figure 24 Real-World Regulation GRL Diagram

6.2. Linked Intentional Elements

When creating views, we need to expand a model from a specific intentional element (often a goal). More specifically, we need to be able to select a goal and choose to see its

directly-connected intentional elements in a goal graph. So, we need to add new intentional elements and their links around the initial goal.

In the original jUCMNav, for creating a new smaller GRL diagram from the main GRL diagram, first we drag and drop the desired goal from the Outline view, or copy and paste it from the main diagram, and then repeat for related elements. This approach is very time consuming. Multiple connected elements can be copy/pasted at once, but they are often hard to see and select properly, leading to incomplete (i.e., erroneous) views. Hence, we felt the need for a new feature to enable us to do this automatically and efficiently.

A new feature was developed to let the user select the desired intentional element for expansion. Using a contextual menu, the tool can add all the other intentional elements connected to it (via contribution, dependency, or decomposition links) in the same diagram. From the tool perspective, new graphical references to these model element definitions are added to the diagram.

As an ongoing example, Figure 25 shows a typical GRL graph. We intend to select a node and try to expand it in a new diagram, using different depths.

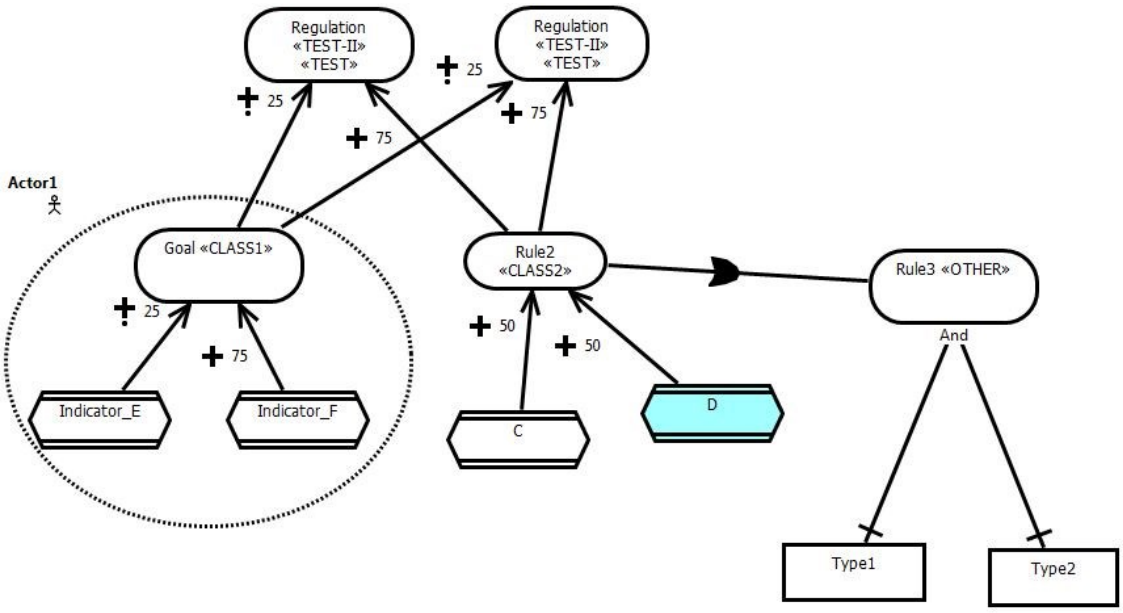


Figure 25 Sample GRL Graph

We select the goal node named *Rule2* and we try to expand it in a new diagram. Figure 26 shows how we try to expand it in *one level deep*. Different depth levels are offered in the contextual menu.

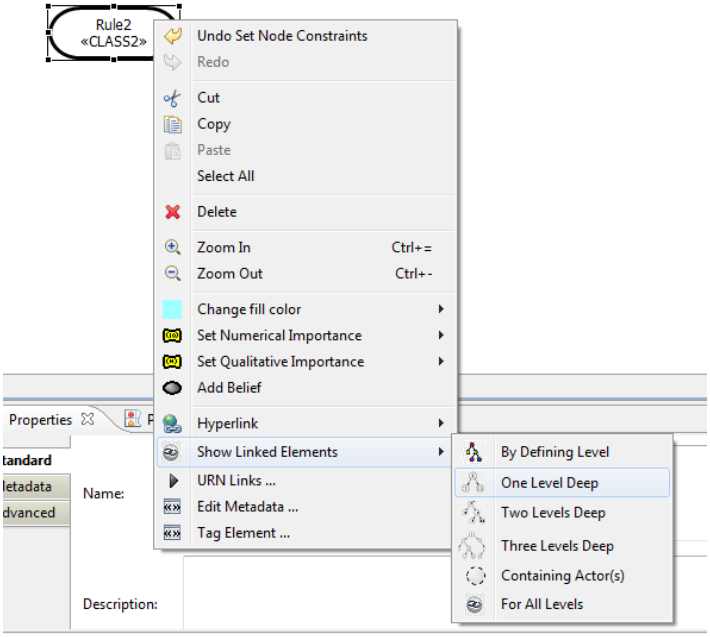


Figure 26 Expanding a GRL Node in One Level

The result of this expansion is shown in Figure 27.

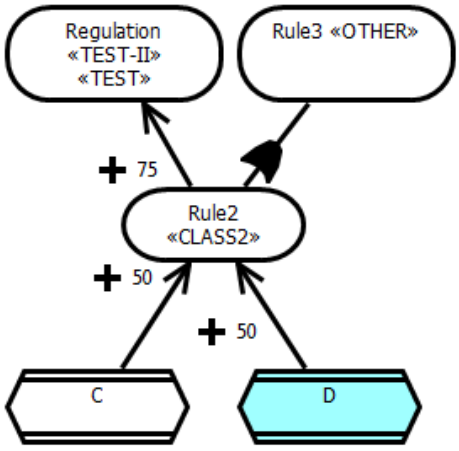


Figure 27 One Level Expanded GRL Graph

Note that there are two different references to the *Regulation* goal in the main GRL graph in Figure 25. However, only one of them is shown in the view of Figure 27. The reason is that this feature only shows the intentional elements that are connected together at the model level. URN and jUCMNav separate model element definitions from their references in diagrams. A definition can be referenced many times (in one or many diagrams). Changes to the definition are reflected to all references automatically. In the example, the two *Regulation* goals are hence simply two references of one intentional element definition named *Regulation*. Figure 28 further illustrates this behaviour of jUCMNav with links, where contribution definitions to the *Regulation* goal are also replicated with many instances.

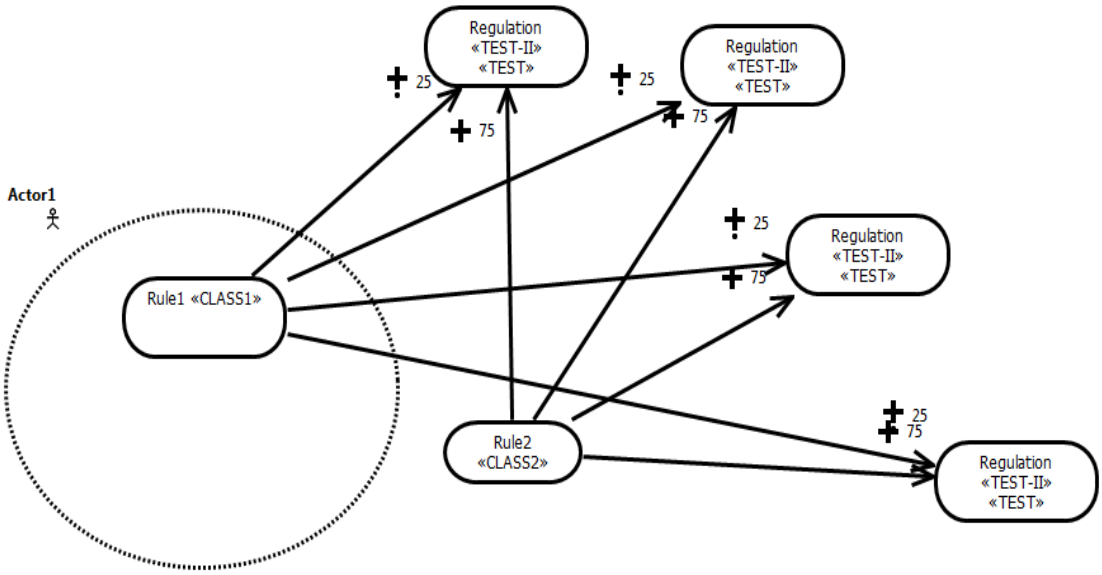


Figure 28 Sample GRL Diagram with Multiple Element/Link References

It must be noted that this expansion feature works with any type of intentional element. Any *goal, softgoal, task, condition* or *indicator* can be selected for expansion.

Moreover, this feature comes in several other variants, for usability and efficiency:

- **Two Levels Deep:** This feature is basically the same as the base feature, but with a depth of 2 (elements connected to elements connected to the original one).
- **Three Levels Deep:** Similar to the previous feature, but with a depth of 3.

- **By Defining Level:** This feature allows any specific depth to be used (1, 2, 3, 4, 5…) for expansion. One more dialog window is required here. The selection of this feature is shown in Figure 29.

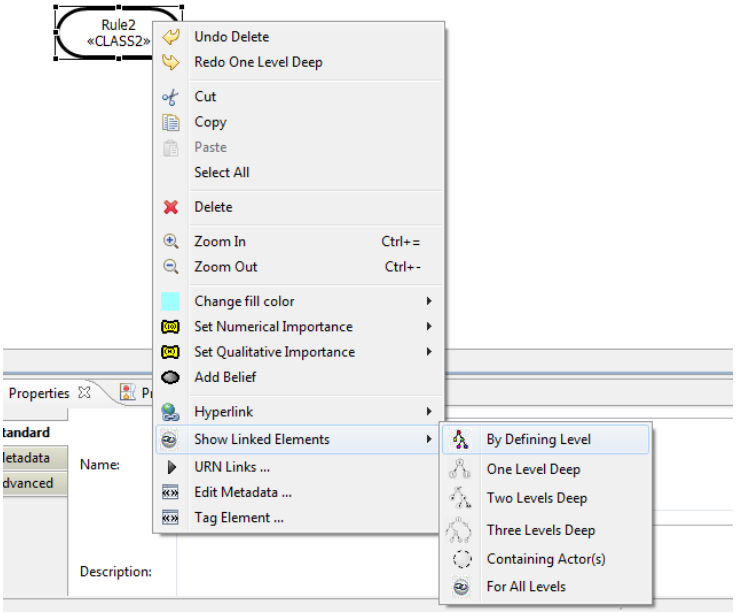


Figure 29 Expansion for Any Level

After selecting this specific feature, a window pops up (Figure 30) to enquire about the depth level of the expansion. Validation is done on the input, and this transformation can also be cancelled.

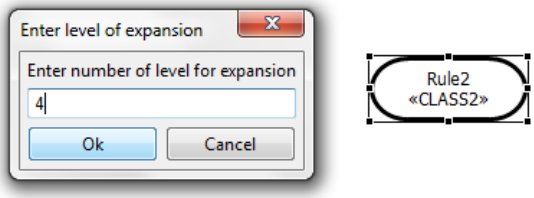


Figure 30 Pop-up Window for Expanding at Any Depth Level

- **All Levels:** This feature is developed with very different assumption. In this feature, it was assumed that we may have many small to relatively average GRL graphs in the model and that we want to expansion all the elements tran-

sitively connected to the initial element in one GRL diagram. Transitive closure is hence used here to compute linked elements.

One issue that was common among all variants of this feature was the redundancy of connected GRL elements. For instance, in a full expansion, a GRL node can have node *g1* and *g2* as its neighbors and node *g1* can have *g2* as its neighbor as well. In such cases, node *g2* must not be shown again. This is handled by the expansion features. Note also that the expansion features can be invoked from an existing graph (with many nodes) and that existing nodes will not be duplicated.

All these features can be accessed from a contextual menu in the jUCMNav environment. By selecting an intentional elements and right-clicking on it, we can see an action provider named *Show Linked Elements*. By hovering over it, we can see all the options provided for expansion. Figure 31 shows this list, which covers the options studied so far, plus two more related to *incoming* links discussed below. The options related to actors are covered in the next sections.

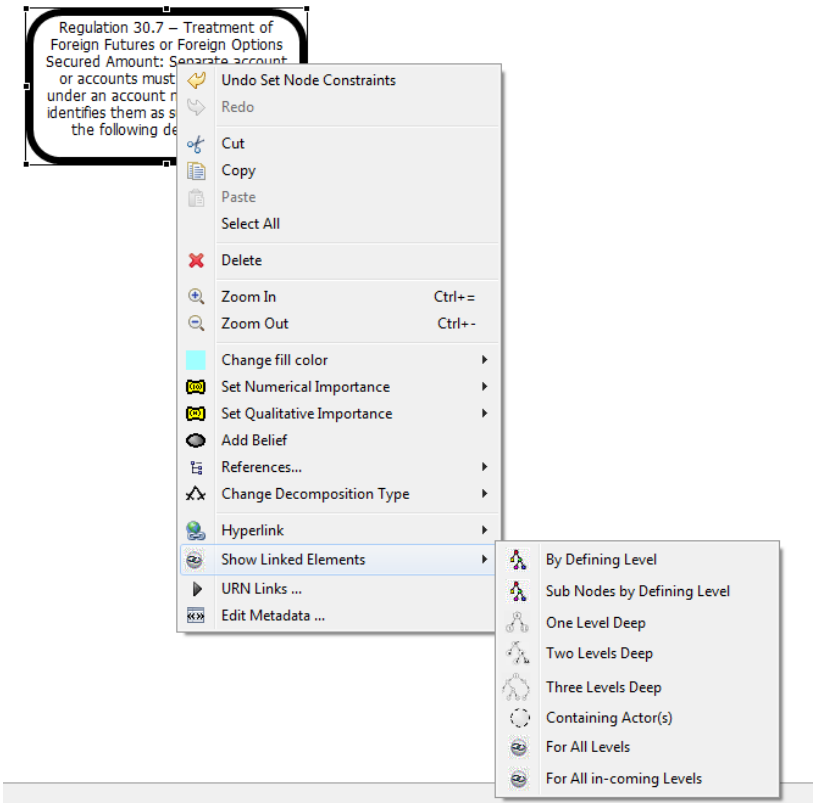


Figure 31 Expansion Menu

- **All Incoming Levels** and **Sub Nodes**: Using these features, users can expand from a node in the direction of its incoming links only. This is very useful for tree-like structures such as the regulation GRL models discussed here. The first one provides a full expansion (all levels), but only through incoming links. The second one, which is a bit more flexible, allows the user to define the desired depth level. As an example, starting form one node in Figure 32, after selecting the proper expansion option, the expanded diagram will be the one in Figure 33.

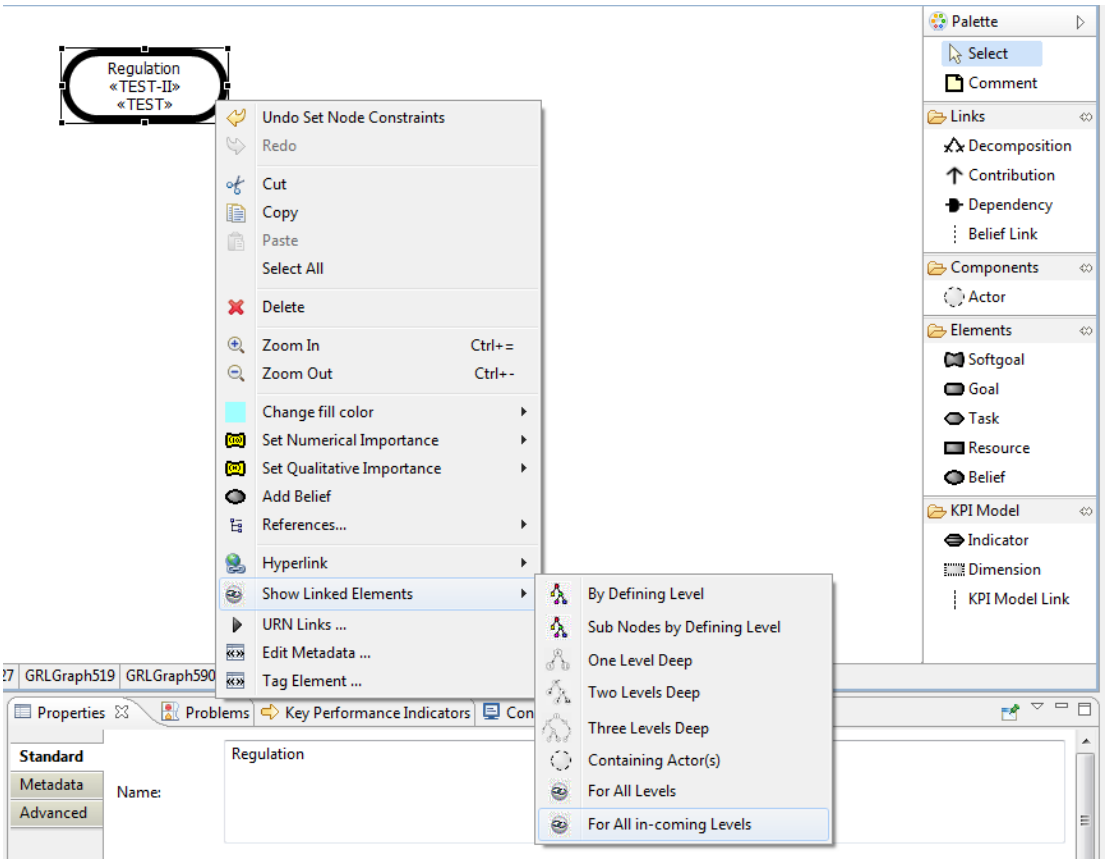


Figure 32 Expansion in the Direction of Incoming Links

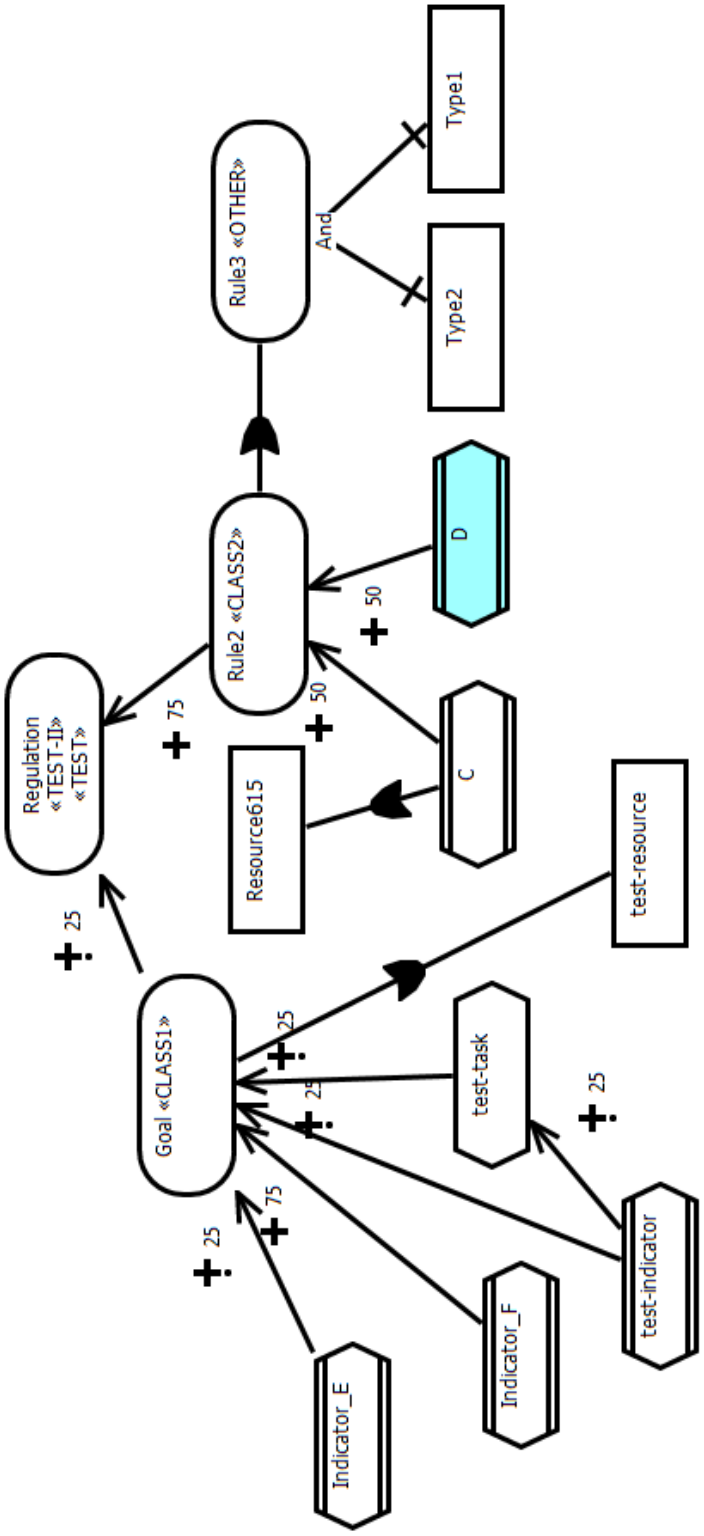


Figure 33 Result of Expansion in the Direction of Incoming Links for Figure 32

6.3. Elements in Actors

In many cases during requirement engineering modelling using jUCMNav, we need to see which intentional elements an actor contains. This is useful as this information might be spread across many diagrams of a GRL model. Conventionally, the user needs to move between diagrams to find out which GRL nodes are placed inside a given actor. More experienced users will look at the *Outline* window in jUCMNav, find the actor's definition, and subsequently its contained intentional elements. However, none of these approaches is convenient and efficient.

Our new jUCMNav feature lets users select an actor and query the model to show its contained intentional elements in an organized manner. For example, Figure 25 and Figure 34 show how an actor can contain multiple intentional elements in different goal graphs (models).

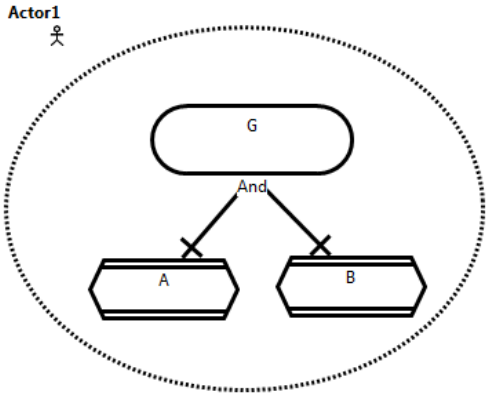


Figure 34 Same Actor Containing other Elements in a Different Model

When a user needs to find out which intentional elements are being contained by an actor in a new GRL diagram, he/she now uses the new feature to see all the contained intentional elements in one place. This feature is easily accessible from a contextual menu in jUCMNav. Figure 35 shows how to invoke that feature by selecting an actor and right-clicking on *Show Bound Intentional Elements*.

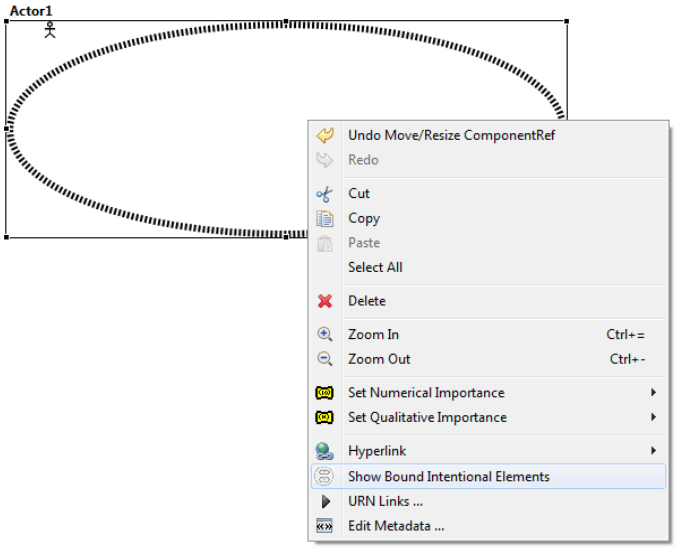


Figure 35 Accessing the Showing Actor's Elements Feature

The end result of using this feature is shown in Figure 36.

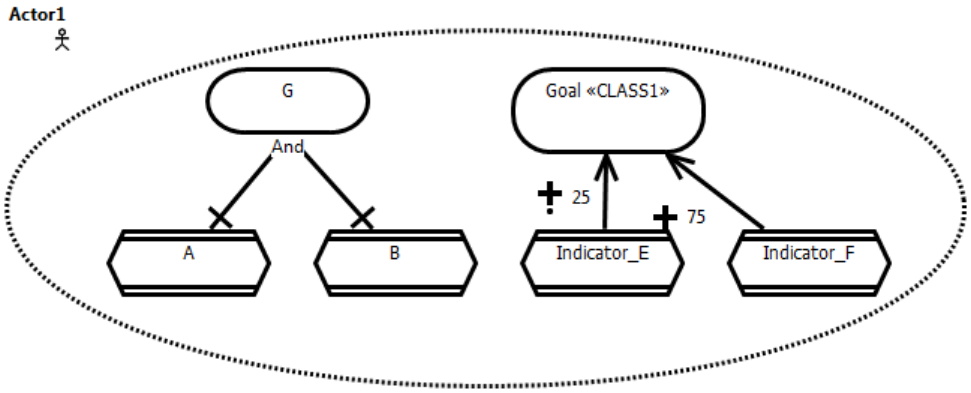


Figure 36 Results of Using Showing Actor1's Elements for Figure 25 and Figure 34

One of the issues in developing this feature was that one intentional element in its containing actor shown in multiple GRL diagrams must not cause the duplication of intentional elements in the new GRL diagram. As was discussed in the previous section, the intentional elements that are repeated in different diagrams inside the same actor are just different references to one single definition. They just reflect the semantics between an actor and its contained element, which is *this GRL element is bound to this actor.* In this feature, only one reference is shown to depict this semantic.

Although, there is no actor support when importing regulations in the main contribution of this thesis, this feature can become highly used after the import and a simple allocation to actors. It is also very useful generically for GRL models outside the regulation domain.

6.4. Containing Actors

In many cases, users need to see to which actor an intentional element is bound, if any. Usually in GRL models, an intentional element is not necessarily visible in all the references of its bound actor. Moving from one diagram to another to find out whether there is a containing actor is hence a time-consuming and error-prone task by nature.

To overcome this issue, a new feature was developed to provide users with a way to visualize, in the current diagram, the containing actor(s) of an intentional element. Again, this feature can be selected in jUCMNav through a contextual menu. As an example, following Figure 37 and Figure 38 show different actors containing the desired GRL node. Notably in Figure 38 there are two actors containing one intentional element and there could be more actors as well. This apparent inconsistency is not standard GRL, but this is tolerated by jUCMNav. The desired element for which we want to show containing actors in a new GRL diagram is *Goal*.

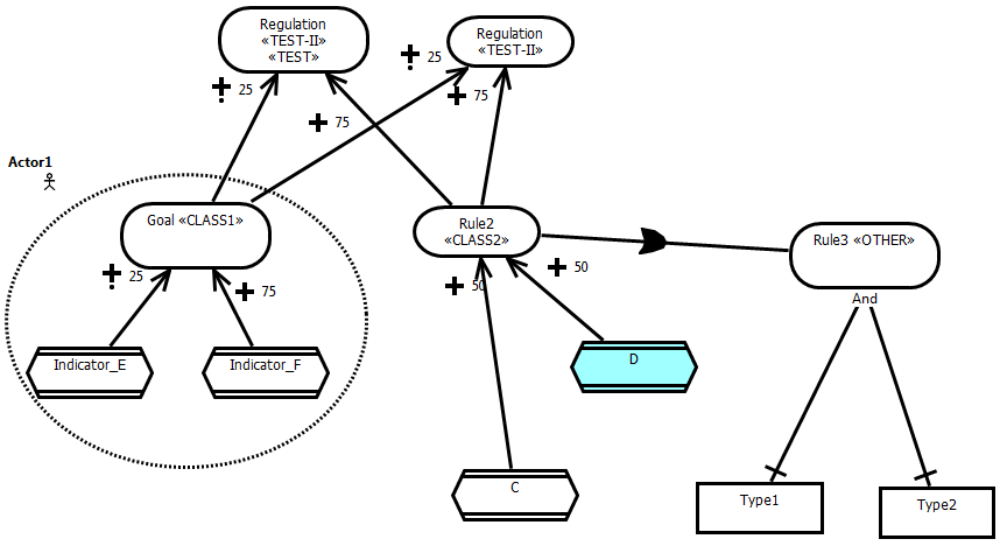


Figure 37 Sample GRL Diagram with Element(s) Contained by an Actor

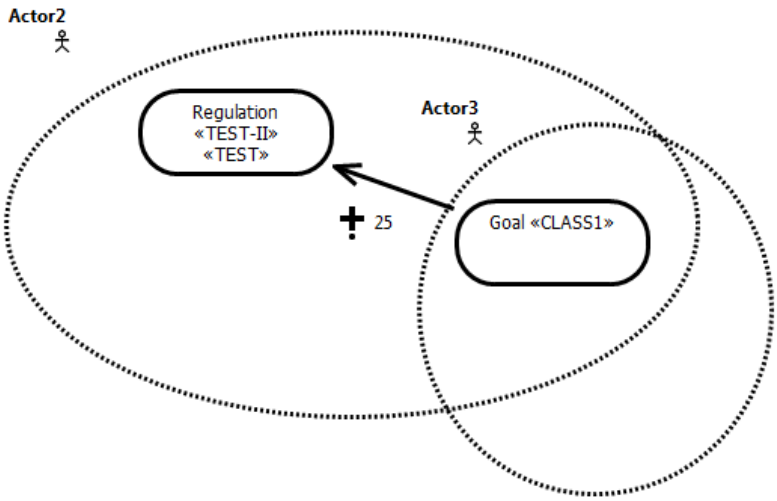


Figure 38 Sample GRL Diagram with Element(s) Contained by multiple Actors

Following the selection of an element in a GRL diagram, the user needs to right-click on the element and select the menu item named *Show Linked Element(s)*. Then, he/she needs to select the correct action (*Containing Actor(s)*) to use the feature.

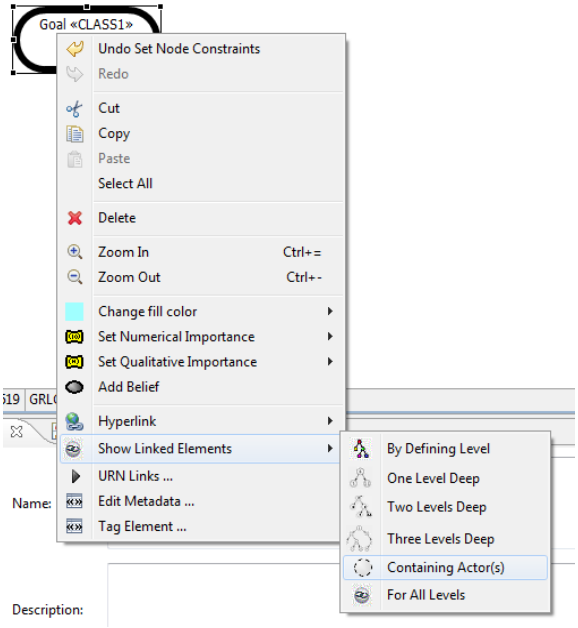


Figure 39 Applying Show Containing Actors Feature

The final result of the application of this feature is illustrated in Figure 40.

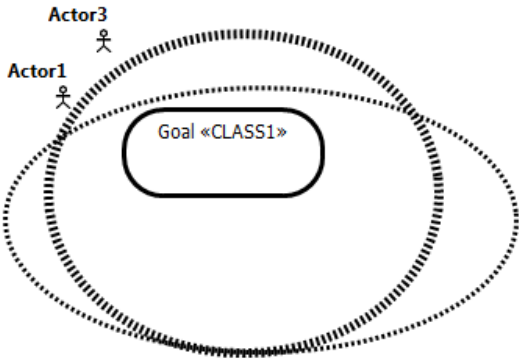


Figure 40 Result of Applying the Show Containing Actor(s) Feature

As can be seen here, only two of the three actors are shown (Actor2 is not shown). The reason for it is that in jUCMNav in each GRL diagram each of the intentional element references only can be contained by one actor only. This means that an intentional element can only be contained by one actor in each GRL diagram.

6.5. Summary

This chapter introduced three new features that were developed for manipulating GRL diagrams in jUCMNav. These features help user to create and tailor GRL diagrams/views to improve understanding. These features were released publicly in jUCMNav 5.4.0. They were tested thoroughly using jUnit (to be discussed in Section 8.1.1) and passed all test cases successfully.

Considering this chapter and the previous ones, at this point GRL models can created from regulation and KPI mappings CSV files and two import mechanisms, and views can easily and efficiently be created using three new jUCMNav features.

In the next chapter, we evaluate the contributions of this thesis and their applications in several real-world case studies.

Chapter 7. Case Studies

In this chapter, three different case studies from different application domains (banking, healthcare privacy, and aerodrome security) are introduced. Each case study introduces new situations and demonstrates how the approach and tools of this thesis can be used in practice.

7.1. Banking Domain Case Study

This case study is related to the banking domain, and is concerned with Commodity Futures Trading Commission (CFTC) Regulation §1.25: *Investment of Customer Funds*. This case study was proposed by an independent consultant from the USA interested in our approach.

7.1.1 Methodology

The regulator provided a rather a simple regulation text that includes main regulations and sub-regulations in different sections. In this case study, the number of regulations is rather small, but little information was provided for the regulations indicators (so we had to invent KPIs for measure compliance in this case study). The approach introduced in Chapter 4 was used to create a regulation CSV file and then an import took place to create the goal model in jUCMNav.

Overview of the Document

There are 28 main regulations in Regulation §1.25, however they all have many sub-regulations. There are no KPIs provided for these regulations, but the latter all have IDs and section numbers. It is obvious that sufficient information for regulations is not provided by the regulator for compliance assessment (this information being probably secret). A snapshot of the source document that was given to us is provided in Figure 41.

	A	B
1	Regulation 1.25 - Investment of Customer Funds	
2	**No**	**Rule Descirption**
3	1	**Permitted Investments** - US Government Securities, US Agency obligations,
4	2	**Asset-based concentration limits** for the total assets held in segregation - No limit -
5	3	**Asset-based concentration limits** for the total assets held in segregation - 10% - MMMFs less than $1 billion in assets and/or have a management company
6	4	**Issuer-based concentration limits** for the total assets held in segregation - 25% Securities of any single issuer of US agency obligations
7	5	**Issuer-based concentration limits** for the total assets held in segregation - 25% Interests in any single family of MMMFs
8	6	**Counterparty concentration limits** for the total assets held in segregation - 25% from a single counterparty
9	7	**Time to maturity** - <24mths except MMMFs
10	8	**Time to maturity** - One day maturity if: instrument deposited on an overnight basis, instrument the FCM owns or has unqualified right to pledge, prices the instrument each day based on current mark to market, DCO reduces assigned value
11	9	**Investments in Instruments issued by affiliates:** - Investing customer funds in obligations of an entity affiliated with the FCM is prohibited. Affiliates include parent company, ultimate holding company, subsidiaries to the lowest level, companies under common ownership. - Investments in affiliate funds is allowed
12	10	**Recordkeeping for each day report for each type of investment** - Type of instruments customer funds have been invested
13	11	**Recordkeeping for each day report for each type of investment** - Original cost of
14	12	**Recordkeeping for each day report for each type of investment** - Current market value of the instruments

Figure 41 A Snapshot of a Banking Domain Regulation Document

This piece of regulations actually belongs to section number *1.25*. So, all the main and sub-regulations in this section have received section numbers that are based on the original section number of this piece of regulation. Although, these section numbers were not provided explicitly by the regulator, it is a valid assumption to give section numbers to all the regulations with respect to their position in the provided document.

There is no alternative description for any of the regulations (they are in English only), but we have translated them to German (as the consultant spoke German) to test

this import capability. There was no URL link provided either but we were able to find the online version on the US Government web site[5].

Problem with Main and Sub-Regulations

In this case study, there are some main regulations presented with an ID, but they often include some sub-regulations without IDs, like in the case of regulation #9 in Figure 41. In some cases, one main regulation is separated based on its concerns into multiple regulations and these regulations have received different IDs in each row of the source regulations document, like in the case of regulations #2 and #3. Thus, modifications were needed to create a proper document structure for the CSV file (and actually to better reflect the source of that regulation).

Problem with Lack of Information

There is no URL for the provided regulation, so additional effort took place to find proper URLs. Additional IDs for sub-regulations and alternative names were also provided manually.

In term of KPIs, as nothing was provided, KPIs and relative weights (of equal values) were introduced by inspecting regulations closely before the creation of CSV file. However, these KPIs and weights were only introduced to demonstrate and test this part of the import filter; they may not be sufficient or correct. A committee of experts would be needed to do this part of the model properly, but this was outside the scope of this experiment.

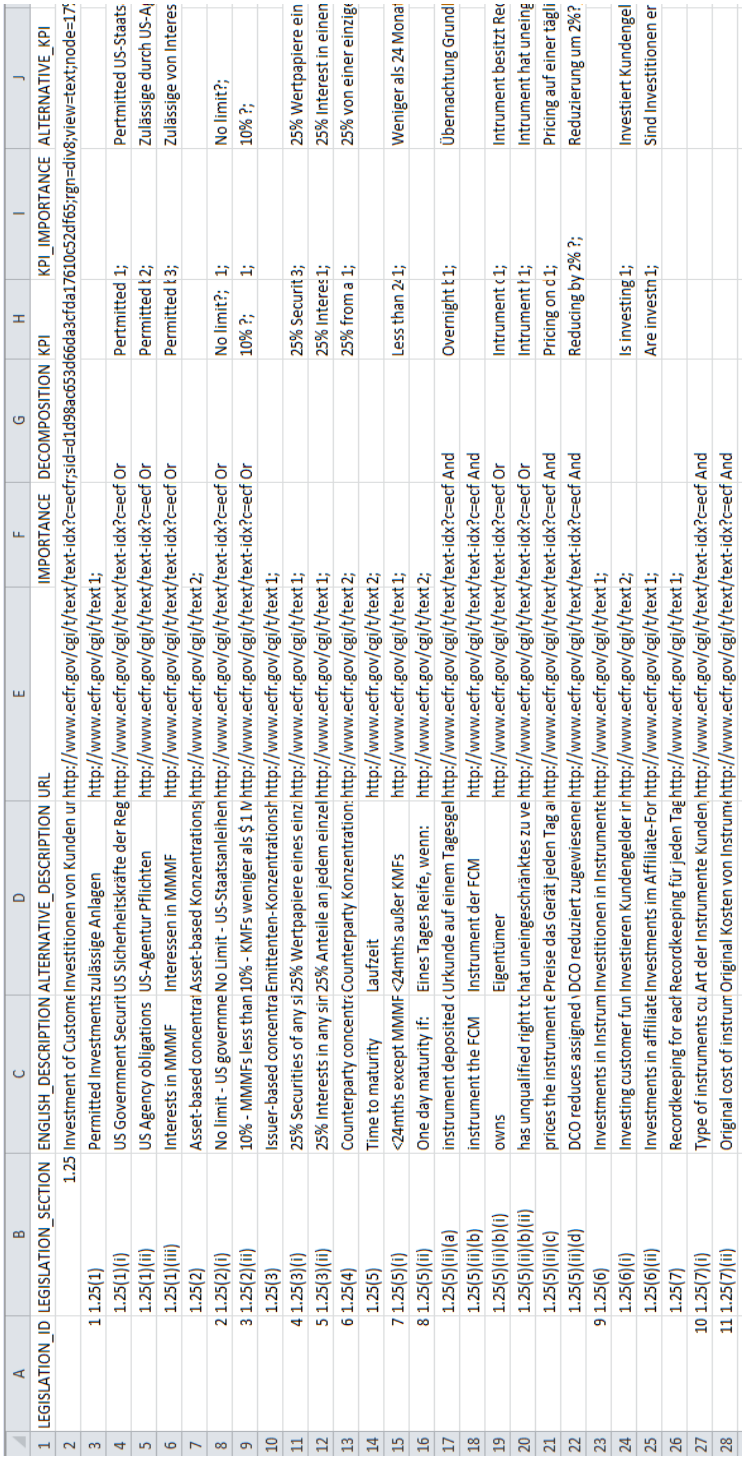
CSV Files Prepared

A well-structured CSV file (Figure 42) was created that contains over 80 rows of regulations and sub-regulations. Also, there are some optional fields (including URLs) that were provided to make a better and more precise goal model. Some KPI conversions got introduced in a second CSV file to demonstrate their functionality. Decompositions and

---

[5] http://www.ecfr.gov/cgi-bin/retrieveECFR?gp=&SID=
3f74582aeb446f8bd4d96a1d37d60085&r=SECTION&n=17y1.0.1.1.1.0.4.24 (October 2012)

contributions between regulations were also taken care of carefully. In this case study, there are no stereotypes provided for the regulations or KPIs.



	A	B	C	D	E	F	G	H	I	J
1	LEGISLATION_ID	LEGISLATION_SECTION	ENGLISH_DESCRIPTION	ALTERNATIVE_DESCRIPTION	URL	IMPORTANCE	DECOMPOSITION	KPI	KPI_IMPORTANCE	ALTERNATIVE_KPI
2		1.25	Investment of Custome	Investitionen von Kunden ur	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecfr;sid=d1d98ac653d66da3cfda17610c52df65;rgn=div8;view=text;node=17					
3	1	1.25(1)	Permitted Investments	zulässige Anlagen	http://www.ecfr.gov/cgi/t/text 1;					
4		1.25(1)(i)	US Government Securit	US Sicherheitskräfte der Reg	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	Or		Pertmitted 1;		Pertmitted US-Staats
5		1.25(1)(ii)	US Agency obligations	US-Agentur Pflichten	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	Or		Permitted l 2;		Zulässige durch US-A
6		1.25(1)(iii)	Interests in MMMF	Interessen in MMMF	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	Or		Permitted l 3;		Zulässige von Interes
7		1.25(2)	Asset-based concentra	Asset-based Konzentrations	http://www.ecfr.gov/cgi/t/text 2;					
8	2	1.25(2)(i)	No limit - US governme	No Limit - US-Staatsanleihen	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	Or		No limit?; 1;		No limit?;
9	3	1.25(2)(ii)	10% - MMMFs less than	10% - KMFs weniger als $ 1 M	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	Or		10% ?; 1;		10% ?;
10		1.25(3)	Issuer-based concentra	Emittenten-Konzentrationsh	http://www.ecfr.gov/cgi/t/text 1;					
11	4	1.25(3)(i)	25% Securities of any si	25% Wertpapiere eines einz	http://www.ecfr.gov/cgi/t/text 1;			25% Securit 3;		25% Wertpapiere ein
12	5	1.25(3)(ii)	25% Interests in any sir	25% Anteile an jedem einzel	http://www.ecfr.gov/cgi/t/text 1;			25% Interes 1;		25% Interest in einen
13	6	1.25(4)	Counterparty concentra	Counterparty Konzentration:	http://www.ecfr.gov/cgi/t/text 2;			25% from a 1;		25% von einer einzig
14		1.25(5)	Time to maturity	Laufzeit	http://www.ecfr.gov/cgi/t/text 2;					
15	7	1.25(5)(i)	<24mths except MMMF	<24mths außer KMFs	http://www.ecfr.gov/cgi/t/text 1;			Less than 24 1;		Weniger als 24 Mona
16	8	1.25(5)(ii)	One day maturity if:	Eines Tages Reife, wenn:	http://www.ecfr.gov/cgi/t/text 2;					
17		1.25(5)(ii)(a)	instrument deposited c	Urkunde auf einem Tagesgel	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	And		Overnight t 1;		Übernachtung Grundl
18		1.25(5)(ii)(b)	instrument the FCM	Instrument der FCM	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	And				
19		1.25(5)(ii)(b)(i)	owns	Eigentümer	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	Or		Intrument c 1;		Intrument besitzt Re
20		1.25(5)(ii)(b)(ii)	has unqualified right tc	hat uneingeschränktes zu ve	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	Or		Intrument l 1;		Intrument hat uneing
21		1.25(5)(ii)(c)	prices the instrument e	Preise das Gerät jeden Tag a	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	And		Pricing on d 1;		Pricing auf einer tägli
22		1.25(5)(ii)(d)	DCO reduces assigned \	DCO reduziert zugewiesene	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	And		Reducing by 2% ?;		Reduzierung um 2%?
23	9	1.25(6)	Investments in Instrum	Investitionen in Instrumente	http://www.ecfr.gov/cgi/t/text 1;					
24		1.25(6)(i)	Investing customer fun	Investieren Kundengelder in	http://www.ecfr.gov/cgi/t/text 2;			Is investing 1;		Investiert Kundengel
25		1.25(6)(ii)	Investments in affiliate	Investments im Affiliate-For	http://www.ecfr.gov/cgi/t/text 1;			Are investn 1;		Sind Investitionen er
26		1.25(7)	Recordkeeping for eact	Recordkeeping für jeden Tag	http://www.ecfr.gov/cgi/t/text 1;					
27	10	1.25(7)(i)	Type of instruments cu	Art der Instrumente Kunden	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	And				
28	11	1.25(7)(ii)	Original cost of instrum	Original Kosten von Instrume	http://www.ecfr.gov/cgi/t/text/text-idx?c=ecf	And				

Figure 42 A Snapshot of Created CSV File

Two KPI conversions (test1 and test2) assigned to four KPIs were also used (Figure 43).

85	KPI_STEREOTYPES	
86	No limit?	#test1
87	10% ?	#test1
88	25% Securities of	#test2
89	25% Interests in a	#test2

Figure 43 Assignment of KPI Conversion to KPIs

In a separate CSV file (Figure 44), all the KPI conversions are defined with the structure introduced in Section 5.2.

	A	B	C
1	RealWorldLabel-test1	Evaluation	exceeds
2	0	0	n
3	1	20	n
4	2	40	n
5	3	100	n
6	4	100	y
7	RealWorldLabel-test2	Evaluation	exceeds
8	0	0	n
9	1	10	n
10	2	30	n
11	3	80	n
12	4	100	y
13	RealWorldLabel-test3	Evaluation	exceeds
14	0	0	n
15	1	100	y

Figure 44 Sample Conversion CSV File Used in This Case Study

There are three conversions defined in Figure 44, named test1, test2 and test3. The third one is a binary conversion that remained unused by the regulation CSV file. Such conversions can be grouped into reusable libraries, and so they do not always all have to be used.

7.1.2 Import of CSV Files and Generating Goal Model

Following the creation of regulation and KPI conversion CSV files, the import can take place. As explained in Sections 4.4.3 and 5.3, an import can take place in two ways. In the first way, the user can import the regulation CSV file alone whereas in the second approach, when there is a KPI conversion CSV file available, the user needs to import this conversion file before the regulation file. The second way was used in the three case studies, including this one. Then, a goal model is generated, with KPI conversions properly assigned. The generated GRL model (too large to be read) is depicted in Figure 46.

To check whether the assignment of KPI conversions has taken place correctly, properties of those KPIs must be checked. For those KPIs to be easily recognizable, their evaluation values (*RealWorldLabel*) was set to '2' in Figure 45. Different KPIs may belong to different conversion groups, the value '2' for *RealWorldLabel* is translated differently for them.

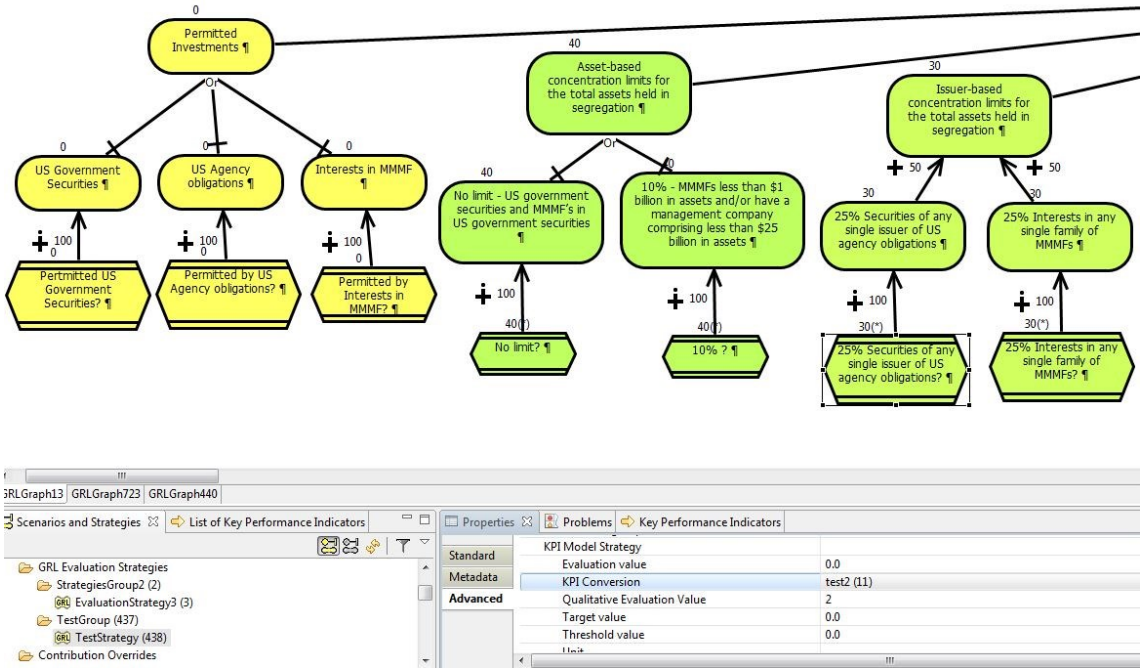


Figure 45 Assignment of KPI Conversion to KPI

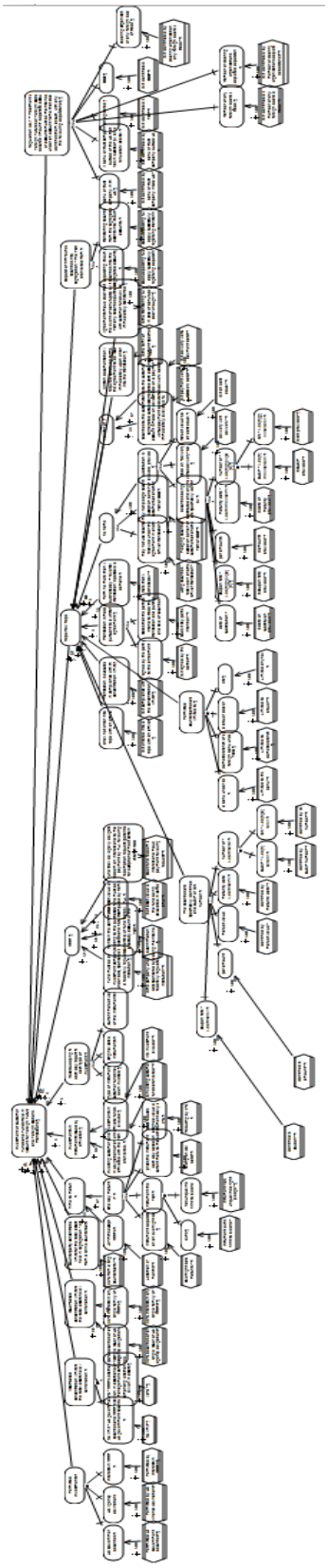


Figure 46 Generated GRL Model for Banking Domain (Not Meant to Be Read)

As can be observed, for two of them the label '2' is translated to '30' (conversion group *test2*) and for the other two it is translated to '40' (conversion group *test1*). These elements are both highlighted in light green in jUCMNav (when the [-100..100] scale is used). Figure 45 also shows one of the KPIs and its properties. Properties of the KPI named "*25% Securities of any single issuer of US agencies obligations?"* are shown and specifically its *KPI Conversion* field is highlighted.

In conclusion, there was no major difficulty in importing a GRL model from regulations in the banking domain. The missing information is actually something that could have been provided by experts, but this was substituted by fake but realistic information in order to test the mechanisms themselves, which behaved properly. The third case study will use more complete information, with realistic KPIs and metadata, in addition to the creation of views to test the features of Chapter 6. The next case study will however make additional use of conditions.

7.2. HIPAA Privacy Case Study

In this case study, a small part of USA's Health Insurance Portability and Accountability Act (HIPAA) is introduced, modelled and analysed. This case study was selected as it was recently proposed by a third-party as a common case study and "challenge" for many approaches that use requirements engineering notations and techniques for modelling and analysing laws[6]. The results of this common case became a special session of the 6th International Workshop on Requirements Engineering and Law (RELAW 2013, [47]). This case study delivers good evaluation value since this piece of regulations carries more issues and challenges in addition to what the previous case study had provided. In terms of results, amongst the five approaches presented at that RELAW 2013 session, our work resulted in the most complete model.

[6] http://gaius.isri.cmu.edu/relaw/2013/conv-challenges.html

7.2.1 Methodology

The same methodology as for the previous case study is used for this one. After the regulations were studied, the regulation CSV file was generated based on the guidelines introduced in Section 4.3.2. In comparison to the previous case study, this HIPPA example introduces a new challenge: the use of *conditions*.

Overview of the Document

The document provided for this case study (section 164.512(f) of HIPPA, see snapshot in Figure 47) is a real piece of regulation with all the characteristics of regulation text documents. There is a very short description at the beginning and then the regulations are presented in plain English. The most interesting part of this piece of regulations is the fact that it is very well organized, with proper *section IDs* provided for all parts.

Parsing the Document and Creating CSV Files

The necessary fields of the regulation CSV file were filled with the information provided in the regulations document. However, one issue faced here is that there is no (internal) ID defined for any regulation. Still, this is not a fatal issue as this field in the CSV file can be left empty (or could be added later should this information become available), since at the GRL level this is just a metadata field with extra information.

Exceptions and Conditions

There exist some conditions and exceptions in this document, like the one in section *f(2)*. As a part of modelling, they must be dealt with properly to get valid regulation models. As discussed in Section 4.3.2, conditions can be easily addressed in GRL modelling with *Dependency* links to *Resource* intentional elements stereotyped as conditions. Exceptions are a type of condition and can be also shown in the same way as conditions.

Problem with One of the Clauses

Interestingly, the clause located in section *f(3)* is an exception, therefore it cannot be handled and treated in the CSV file as a regular regulation. Also, it cannot have any regulation ID or section ID.

§ 164.512 Uses and disclosures for which an authorization or opportunity to agree or object is not required.

A covered entity may use or disclose protected health information without the written authorization of the individual, as described in §164.508, or the opportunity for the individual to agree or object as described in §164.510, in the situations covered by this section, subject to the applicable requirements of this section. When the covered entity is required by this section to inform the individual of, or when the individual may agree to, a use or disclosure permitted by this section, the covered entity's information and the individual's agreement may be given orally.

…

(f) *Standard: Disclosures for law enforcement purposes.* A covered entity may disclose protected health information for a law enforcement purpose to a law enforcement official if the conditions in paragraphs (f)(1) through (f)(6) of this section are met, as applicable.

(1) *Permitted disclosures: Pursuant to process and as otherwise required by law.* A covered entity may disclose protected health information:

(i) As required by law including laws that require the reporting of certain types of wounds or other physical injuries, except for laws subject to paragraph (b)(1)(ii) or (c)(1)(i) of this section; or

(ii) In compliance with and as limited by the relevant requirements of:

(A) A court order or court-ordered warrant, or a subpoena or summons issued by a judicial officer;

(B) A grand jury subpoena; or

(C) An administrative request, including an administrative subpoena or summons, a civil or an authorized investigative demand, or similar process authorized under law, provided that:

(1) The information sought is relevant and material to a legitimate law enforcement inquiry;

(2) The request is specific and limited in scope to the extent reasonably practicable in light of the purpose for which the information is sought; and

(3) De-identified information could not reasonably be used.

(2) *Permitted disclosures: Limited information for identification and location purposes.* Except for disclosures required by law as permitted by paragraph (f)(1) of this section, a covered entity may disclose protected health information in response to a law enforcement official's request for such information for the purpose of identifying or locating a suspect, fugitive, material witness, or missing person, provided that:

(i) The covered entity may disclose only the following information:

(A) Name and address;

(B) Date and place of birth;

(C) Social security number;

(D) ABO blood type and rh factor;

(E) Type of injury;

(F) Date and time of treatment;

(G) Date and time of death, if applicable; and

(H) A description of distinguishing physical characteristics, including height, weight, gender, race, hair and eye color, presence or absence of facial hair (beard or moustache), scars, and tattoos.

(ii) Except as permitted by paragraph (f)(2)(i) of this section, the covered entity may not disclose for the purposes of identification or location under paragraph (f)(2) of this section any protected health information related to the individual's DNA or DNA analysis, dental records, or typing, samples or analysis of body fluids or tissue.

(3) *Permitted disclosure: Victims of a crime.* Except for disclosures required by law as permitted by paragraph (f)(1) of this section, a covered entity may disclose protected health information in response to a law enforcement official's request for such information about an individual who is or is suspected to

be a victim of a crime, other than disclosures that are subject to paragraph (b) or (c) of this section, if:

(i) The individual agrees to the disclosure; or

(ii) The covered entity is unable to obtain the individual's agreement because of incapacity or other emergency circumstance, provided that:

(A) The law enforcement official represents that such information is needed to determine whether a violation of law by a person other than the victim has occurred, and such information is not intended to be used against the victim;

(B) The law enforcement official represents that immediate law enforcement activity that depends upon the disclosure would be materially and adversely affected by waiting until the individual is able to agree to the disclosure; and

(C) The disclosure is in the best interests of the individual as determined by the covered entity, in the exercise of professional judgment.

(4) *Permitted disclosure: Decedents.* A covered entity may disclose protected health information about an individual who has died to a law enforcement official for the purpose of alerting law enforcement of the death of the individual if the covered entity has a suspicion that such death may have resulted from criminal conduct.

(5) *Permitted disclosure: Crime on premises.* A covered entity may disclose to a law enforcement official protected health information that the covered entity believes in good faith constitutes evidence of criminal conduct that occurred on the premises of the covered entity.

(6) *Permitted disclosure: Reporting crime in emergencies.*

(i) A covered health care provider providing emergency health care in response to a medical emergency, other than such emergency on the premises of the covered health care provider, may disclose protected health information to a law enforcement official if such disclosure appears necessary to alert law enforcement to:

(A) The commission and nature of a crime;

(B) The location of such crime or of the victim(s) of such crime; and

(C) The identity, description, and location of the perpetrator of such crime.

(ii) If a covered health care provider believes that the medical emergency described in paragraph (f)(6)(i) of this section is the result of abuse, neglect, or domestic violence of the individual in need of emergency health care, paragraph (f)(6)(i) of this section does not apply and any disclosure to a law enforcement official for law enforcement purposes is subject to paragraph (c) of this section.

Figure 47 HIPPA § 164.512(f)

Problem with Indicators and their Conversion Groups

In this case study, it is assumed that indicators (KPIs) must be figured out from the regulation document and should be placed in the regulation CSV file to be reflected in the generated goal model. Also, the conversion groups for KPIs need to be defined and then used in the regulations model. Again, without experts available, finding KPIs, conversion groups and relative weights was quite a challenge, and the existence of conditions and exceptions made this process harder. But, as for the first example, we were less interested in the overall validity of the model here than in showing that a model could be generated automatically when such information is available.

CSV Files Prepared

Following the discovery of issues and of ways to mitigate them, the two CSV files were created. There are some fields of the regulation CSV file clearly provided in the regulation document: regulations descriptions and their section IDs, contribution and decomposition links (although relative weights for contributions were estimated), URLs, and conditions.

There are two conversion groups defined in this model; one binary group and one that has four different values (see Table 3). This table is a tabular presentation of the conversion groups for the indicators/KPIs in this case study.

Table 3 Conversion Groups for HIPPA Example

RealWorldLabel-Test1	**Evaluation**	**Exceeds**
No	0	false
Somehow	40	false
Yes	100	false
Yes and also exceeds	100	true
RealWorldLabel-Test2	**Evaluation**	**Exceeds**
No	0	false
Yes	100	false

Once available, these conversion groups can be used in the regulation CSV file to make the generation of the goal model precise. These values must be set under the KPI_STEREOTYPES field for each KPI, in different rows of the CSV file. Figure 48 shows these values set in the proper field for some of these KPIs.

37	KPI_STEREOTYPES	
38	Is the protected information presented as the law request based on certain situation?	#Test2
39	Is it in compliance with a court order or court-ordered warrant, or a subpoena or summons iss	#Test1
40	Is it in compliance with a grand jury subpoena?	#Test1
41	Is the information relevant enough to lawinquiry?	#Test1
42	Is the request specific and limited enough to the extent practicable in light of the purpose?	#Test1
43	Are the de-identified information usable?	#Test2
44	Are name and address presented?	#Test2
45	Are date and place of birth presented?	#Test2
46	Is SSN presented?	#Test2
47	Are ABO blood type and rh factor presented?	#Test2
48	Is type of injury presented?	#Test2
49	Are date and time of treatment presented?	#Test2
50	Are date and time of death presented?	#Test2
51	Is a description of distinguishing of any physical characteristics?	#Test2
52	Agreed to disclosure?	#Test2
53	Does the law enforcement official represent that such information is needed?	#Test2
54	Does the law enforcement official represent that immediate law enforcement activity that de	#Test2
55	Is the disclosure in the best interests of the individual as determined by the covered entity?	#Test2
56	Does the covered entity have a suspicion that such death may have resulted from criminal cor	#Test1
57	Does the covered entity believes in good faith constitutes evidence of criminal conduct that c	#Test1
58	Alert law enforcement to the commission and nature of a crime?	#Test2
59	Alert law enforcement to the location of such crime or of the victim(s) of such crime?	#Test2
60	Alert law enforcement to the identity?	#Test2
61	Alert law enforcement to the description?	#Test2

Figure 48 A Sample of Conversion Groups Set for a few KPIs of the HIPPA Example

An extract of the regulation CSV file is provided in Figure 49. A simple French translation was provided to showcase the fact that bilingual models could be supported. The conditions for each regulation are defined in the KPI_IMPORTANCE column. The importance values (relative contribution weights) between regulations or between regulations and KPIs are considered equal here, for simplicity and because realistic values require the involvement of experts unavailable for this experiment.

Following the creation of two CSV files, the import can be performed by jUCMNav and the regulation GRL model is created.

	A	B	C	D	E	F	G	H	
1	LEGISLATION_ID	LEGISLATION_SECTION	ENGLISH_DESCRIPTION	ALTERNATIVE_DESCRIPTION	URL	IMPORTANCE	DECOMPOSITION	KPI	KPI_IMPORTANCE
2		164.512 (f)	A covered entity may di	Une entité visée peut comm	http://gaius.isri.cmu.edu/relaw/2013/conv-challenges-excerpt.pdf				
3		164.512 (f)(1)	(Pursuant to process anc	Divulgations autorisées: Con	http://gaius.isri.cmu.edu	Or			
4		164.512 (f)(1)(i)	As required by law inclu	Tel que requis par la loi, y co	http://gaius.isri.cmu.edu	Or		Is the prot	#C_except for laws subject to paragraph (b)(1)(ii) or (c)(1)(i) of this section;
5		164.512 (f)(1)(ii)	In compliance with and	En conformité avec et limitée	http://gaius.isri.cmu.edu	Or			
6		164.512 (f)(1)(ii)(A)	A court order or court-or	Une ordonnance du tribunal	http://gaius.isri.cmu.edu	Or		Is it in compliance with a court order or court-ordered warrant, or a subpoena or summons issued by	
7		164.512 (f)(1)(ii)(B)	A grand jury subpoena;	Une citation à comparaître de	http://gaius.isri.cmu.edu	Or		Is it in compliance with a grand jury subpoena?;	
8		164.512 (f)(1)(ii)(C)	An administrative reque	Une demande administrative	http://gaius.isri.cmu.edu	Or			
9		164.512 (f)(1)(ii)(C)(1)	The information sought	L'information demandée est	http://gaius.isri.cmu.edu	And		Is the information relevant enough to lawinquiry?;	
10		164.512 (f)(1)(ii)(C)(2)	The request is specific a	La demande est spécifique e	http://gaius.isri.cmu.edu	And		Is the request specific and limited enough to the extent practicable in light of the purpose?;	
11		164.512 (f)(1)(ii)(C)(3)	De-identified informati	Renseignements dépersonn	http://gaius.isri.cmu.edu	And		Are the de-identified information usable?;	
12		164.512 (f)(2)	(Limited information for	Divulgations autorisées: Peu	http://gaius.isri.cmu.edu	Or		#C_Except for disclosures required by law as permitted by paragraph (f)(1) of this section;	
13		164.512 (f)(2)(i)	The covered entity may	L'entité visée peut divulguer	http://gai 1;				
14		164.512 (f)(2)(i)(A)	Name and address;	Nom et adresse;	http://gaius.isri.cmu.edu	And		Are name and address presented?;	
15		164.512 (f)(2)(i)(B)	Date and place of birth;	Date et lieu de naissance;	http://gaius.isri.cmu.edu	And		Are date and place of birth presented?;	
16		164.512 (f)(2)(i)(C)	Social security number;	Numéro de sécurité sociale;	http://gaius.isri.cmu.edu	And		Is SSN presented?;	
17		164.512 (f)(2)(i)(D)	ABO blood type and rh f	Groupe sanguin ABO et facte	http://gaius.isri.cmu.edu	And		Are ABO blood type and rh factor presented?;	
18		164.512 (f)(2)(i)(E)	Type of injury;	Type de blessure;	http://gaius.isri.cmu.edu	And		Is type of injury presented?;	
19		164.512 (f)(2)(i)(F)	Date and time of treatm	Date et heure de traitement;	http://gaius.isri.cmu.edu	And		Are date and time of treatment presented?;	
20		164.512 (f)(2)(i)(G)	Date and time of death,	Date et heure de la mort, le c	http://gaius.isri.cmu.edu	And		Are date and time of death presented?;	
21		164.512 (f)(2)(i)(H)	A description of distingu	Une description des caractéri	http://gaius.isri.cmu.edu	And		Is a description of distinguishing of any physical characteristics?;	
22		164.512 (f)(2)(ii)	The covered entity may	Sauf dans la mesure permise	http://gai 1;			#C_Except as permitted by paragraph (f)(2)(i) of this section;	
23		164.512 (f)(3)	(Victims of a crime) to a	Divulgation permise: les victi	http://gaius.isri.cmu.edu	Or		#C_Except for disclosures required by law as permitted by paragraph (f)(1) of this section;	
24		164.512 (f)(3)(i)	The individual agrees to	L'individu consent à la divulg	http://gaius.isri.cmu.edu	Or		Agreed to disclosure?;	
25		164.512 (f)(3)(ii)	The covered entity is un	L'entité visée est incapable d	http://gaius.isri.cmu.edu	Or			
26		164.512 (f)(3)(ii)(A)	The law enforcement of	Le responsable de l'applicatio	http://gaius.isri.cmu.edu	And		Does the law enforcement official represent that such information is needed?;	
27		164.512 (f)(3)(ii)(B)	The law enforcement of	Le responsable de l'applicatio	http://gaius.isri.cmu.edu	And		Does the law enforcement official represent that immediate law enforcement activity that depends	
28		164.512 (f)(3)(ii)(C)	The disclosure is in the b	La divulgation est dans l'intér	http://gaius.isri.cmu.edu	And		Is the disc	#C_if the covered entity has a suspicion that such death may have resulted from criminal c

Figure 49 Created CSV File for § 164.512

7.2.2 Generated Regulation Model

In this case study, the first import was related to conversion groups, resulting in an instance of a URN model with conversion groups. The KPI conversion groups created from Table 3 are illustrated in Figure 50.

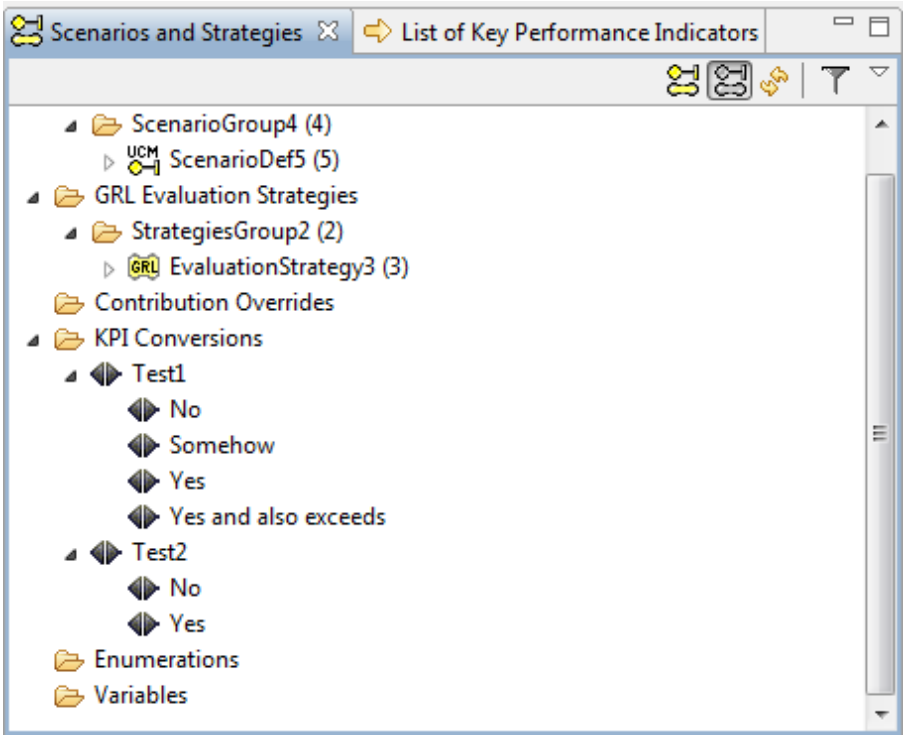


Figure 50 Imported KPI Conversion Groups and Generated URN Instance

The second and main import is related to regulations (which connect KPIs to conversion groups on the fly), resulting in a GRL model whose complete diagram is shown in Figure 51.

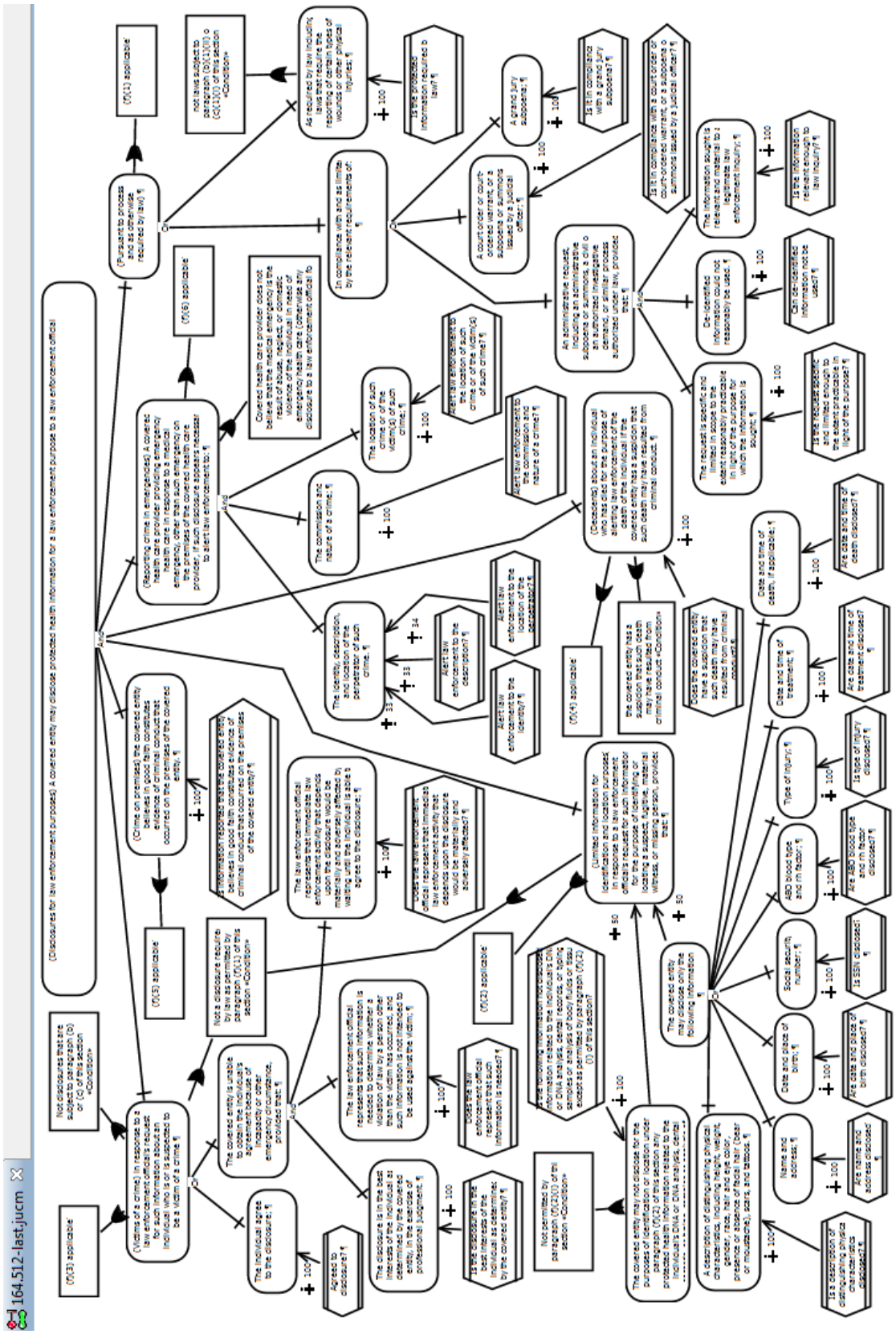


Figure 51 Generated Goal Model for HIPPA § 164.512 (Not Meant to Be Read)

In Figure 51, there are some *dependency* links between some regulations and *Conditions* (stereotyped resource intentional elements). These are used to model the conditions and exceptions that exist for the regulations.

Interestingly, as was mentioned earlier, there is a clause in one of the regulations (*(f)(3)*) that cannot be modelled as a regulation despite being structured in such a way in the document. To be more precise, this clause is expressed in a conditional way, and this is why it cannot be reflected in the regulation model as a goal. It needs to be modelled as a condition to its sibling clause. This issue and the way it is addressed are illustrated in Figure 52. This clause is the condition on the right side of the main regulation node.

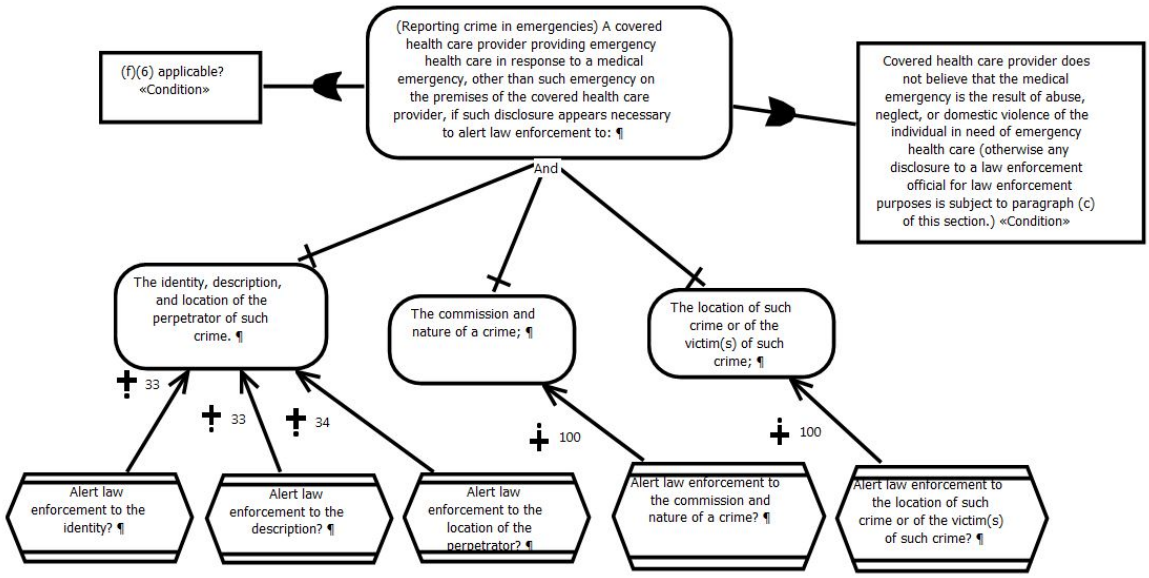


Figure 52 Clause Treated as a Condition in the HIPPA Model

Moreover, there is a main, general condition for all regulations at the highest level of granularity. This condition is *applicability*, discussed in the introduction of the regulation document. Figure 53 shows these conditions and the way they are modelled in the generated goal model. There is an alternative way that could have been used for modelling this condition for all these regulations. We could have introduced one generic condition for *applicability*, and all relevant regulations would have been connected with dependency links to this condition. However, to reduce the complexity of the model and enable the use of more precise conditions, this approach was rejected.

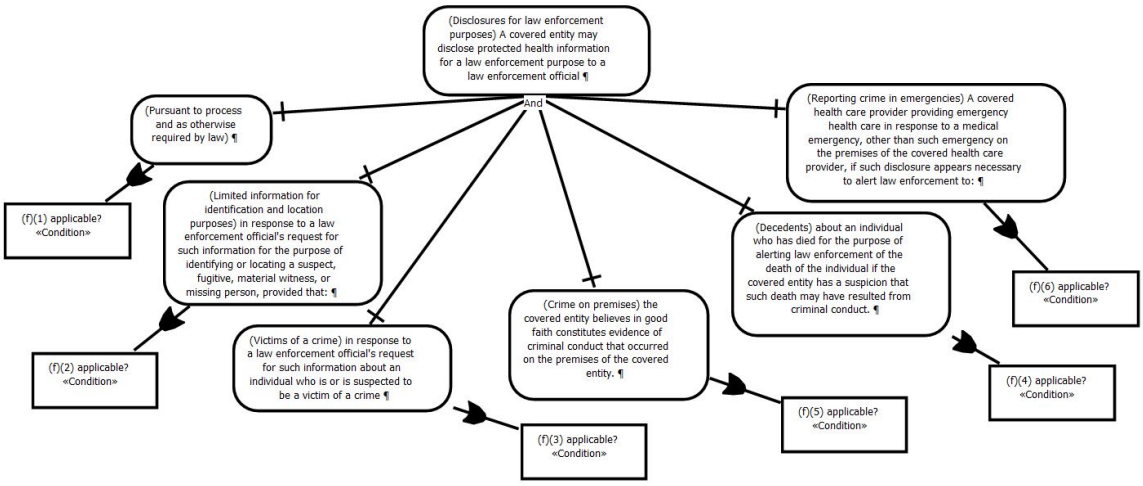


Figure 53 General (Main) Condition for Regulation with Highest Granularity

The conditions are used in the evaluation of models during compliance analysis. For example, in Figure 54, one of the conditions is not met (it is assigned a value of *zero*) and this causes the regulation depending on it *not* to be evaluated, as if this part was disconnected or removed from the model. Such part is hence greyed out automatically by jUCMNav, as visual feedback. To make this work properly in jUCMNav, *conditions* in a GRL model must tagged with a *Condition* stereotype. The built-in evaluation algorithms in jUCMNav then handle conditions properly, according to Shamsaei's approach [53].

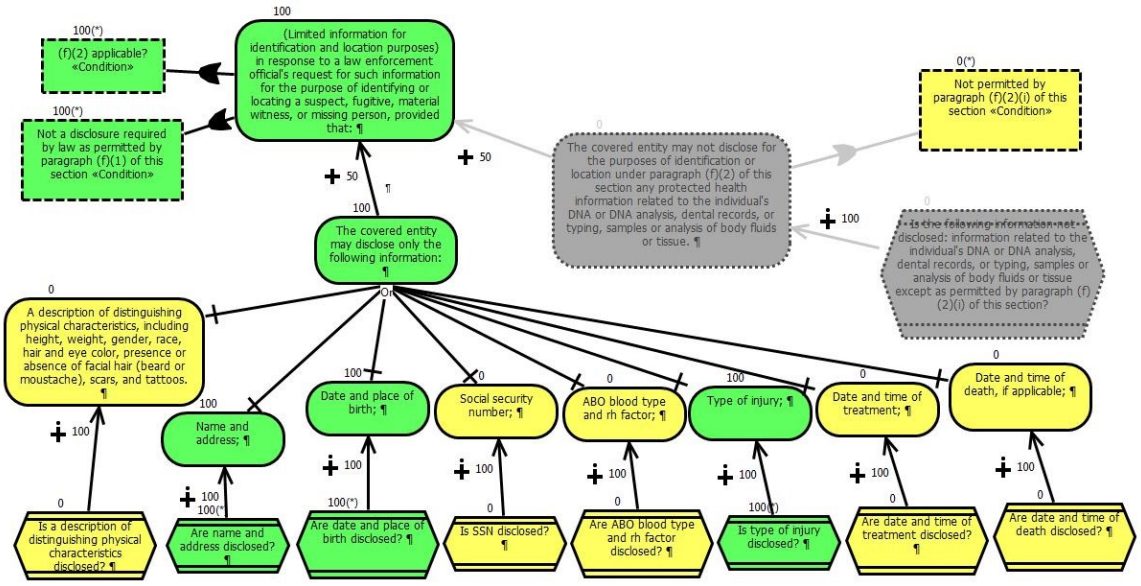


Figure 54 Conditions in Use for § 164.512(f)(2)

In Figure 54, during the analysis where evaluation strategies (reflecting observable values from the real world, fed to KPIs and to conditions) are used, jUCMNav converts KPI values to GRL satisfaction values (according to their respective conversion tables), disconnects the parts of the model whose conditions are not met, and finally propagate satisfaction values to the rest of the intentional elements and to the actors of the models. In this figure, for this strategy, the regulation is satisfied.

7.3. Aerodrome Security Case Study

This third and larger case study comes from yet another domain, this time related to aviation security regulations. This example was developed for a national regulator interested in moving from prescriptive regulations towards outcome-based compliance measurement [61]. Several models covering different regulations were developed as part of a collaborative project but they cannot be discussed in this thesis due to the sensitive nature of aviation security. However, this example is based on regulations that are publicly available[7]. KPIs reflecting inspection questionnaires were available and used in this case study, however these questionnaires are confidential and hence the details of the KPIs themselves will not be discussed. Consequently, only their IDs (not their descriptions) will be used in the model and diagrams. In addition, for each regulation, there could exist multiple KPIs.

7.3.1 Methodology

The regulator provided regulation text in this case as a *Microsoft Excel* file, generated from a regulations database. This file contains nearly thousands of rows of regulations, but this case study was scoped to a particular act (A-2 of the Aeronautics Act), composed of 273 rows of regulations.

---

[7] Aeronautics Act (R.S.C., 1985, c. A-2), Canada, http://laws-lois.justice.gc.ca/eng/acts/A-2/FullText.html (2012-01-01)

An Overview of the Document

For each regulation, there were some fields provided. This information includes legislation ID, legislation descriptions (both in English and French, as the official act is bilingual) and related KPI IDs.

Problem with Repeated IDs

One notable thing about this case study is that not all regulations in this document have one KPI. There are some regulations with multiple KPIs, but they are structured in different rows of the Excel document. Thus, to make each row unique, the regulator gave a unique legislation ID to every single row in the document. As a result, a regulation with multiple KPIs was given different regulation IDs in the legislation document. However, this is not a correct way of presenting the information in the case of multiple KPIs for one regulation. Hence, pre-processing of the document was needed to spot this type of issue and address it properly (with one unique ID per regulation only).

Problem with Uniqueness of Regulation Descriptions

Another issue is related to the descriptions of the regulations. GRL models expect the names of intentional elements to be unique. However, in some cases in this document, regulations were had similar descriptions as the structure of the document contained different *classes* of organizations (aerodromes) for which some of the regulations were duplicated. Yet, those regulations are provided with another complementary description (e.g., different IDs) to make them unique. Renaming was performed to differentiate them, including their sub-clauses.

Created Goal Model

After fixing the regulation CSV file and creating another CSV file with realistic KPI conversion mappings, the import mechanisms were used to create the resulting GRL model, whose overview (not meant to be read) is given in Figure 55. Its complexity can easily be assessed (270 goals and 188 KPIs), and hence different views will be required later to improve understanding (to be addressed in the next section).

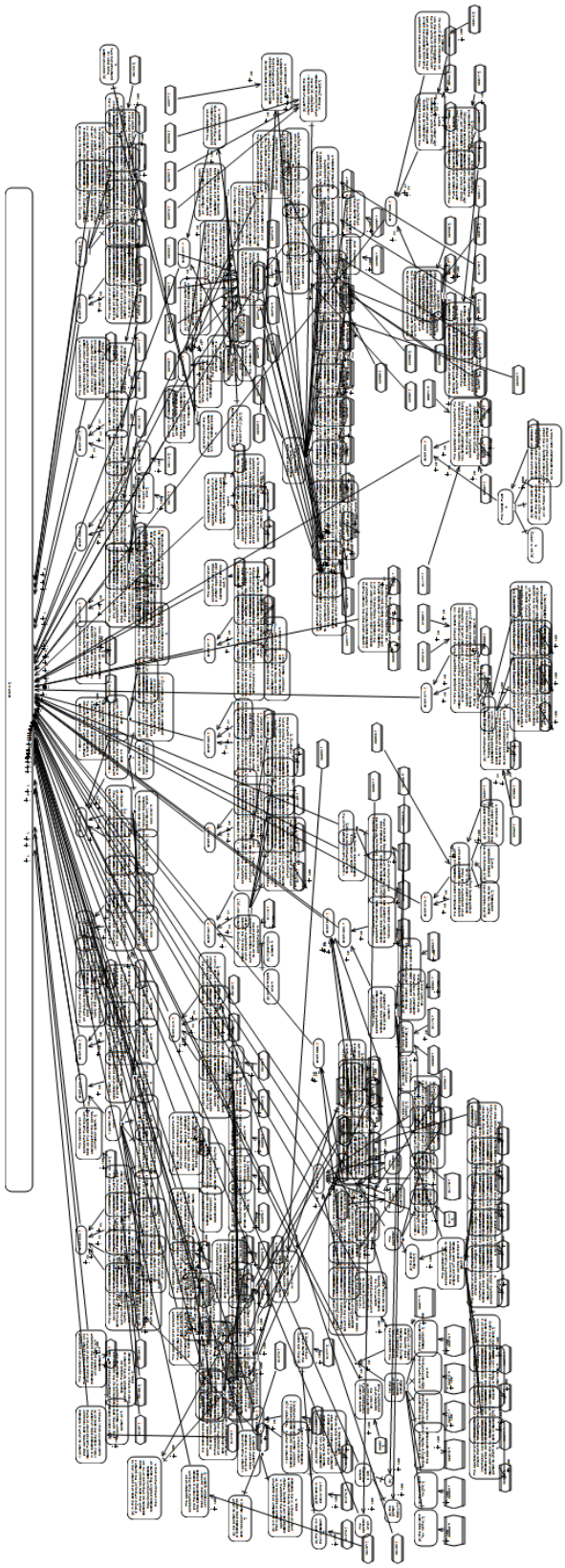


Figure 55 Goal Model Generated Through Import (Not Meant to Be Read)

The relative weights of the contributions are considered equal for simplicity (and because the regulator did not want to share this strategic and confidential information at the time of the experiment). For instance in Figure 56, the three regulations (goals) contributing to the regulation *AA.79* are equally contributing (33, 33, and 34, addressing rounding errors when summing up to 100). We used a similar assumption for KPI weights. For instance, in Figure 56, two KPIs shown as *Question7* and *Question9* are both contributing equally (50 and 50) to the regulation they measure.

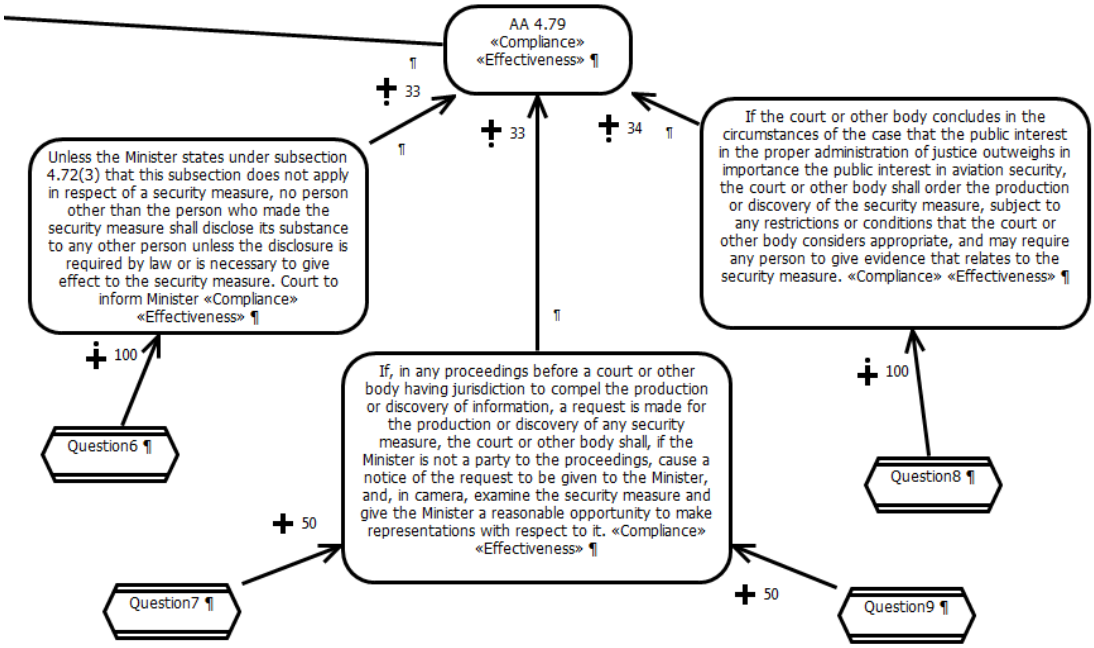


Figure 56 An Example for KPIs with Equal Weights

Moreover, the additional information provided in the CSV file, like the URL, the Alternative_Description, etc. are being stored as metadata of the model elements. An example is shown in Figure 57. In this example, the hyperlink (which can be navigated to), Legislation_ID and Legislation_Section fields stored in the CSV file are shown as metadata for the selected element (with thicker border).

Note also that jUCMNav was used here to display the model in the alternative language (French). In this case, the initial language information (in English in this example) becomes stored in hidden Alternative_Description metadata, ready to be switched back.

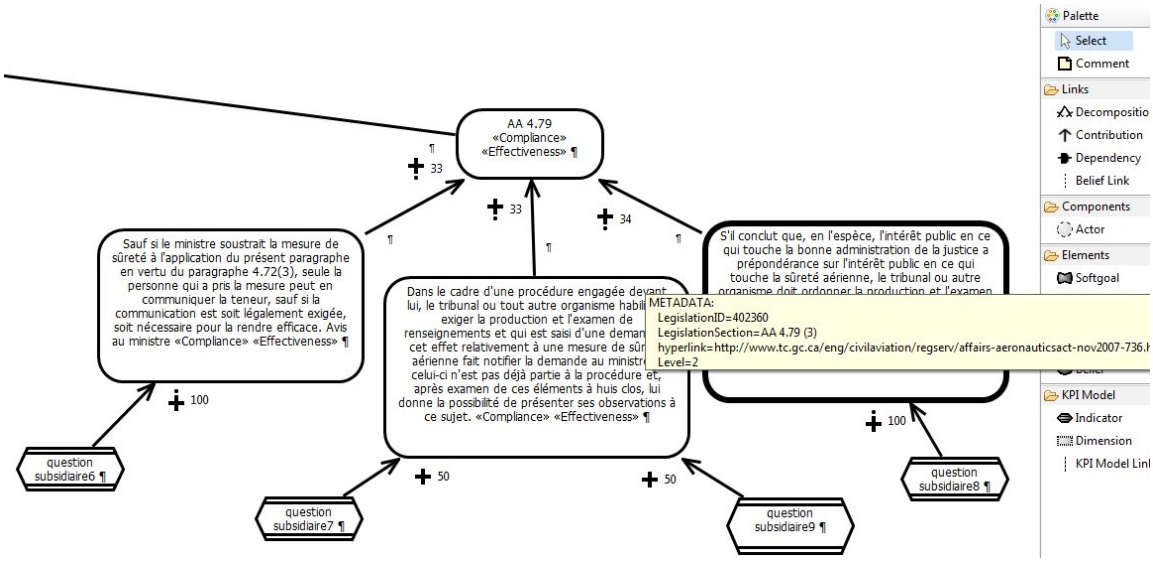


Figure 57 Information Stored as Metadata for Each Element

7.3.2 Manipulation of Diagrams

The model generated following the import results in one huge goal graph with many intentional elements and links between them. For understanding and analysis purposes, we need to be able to show different parts of this goal model as separate and more manageable views. The new jUCMNav features contributed in Chapter 6 were used for that purpose. As an example, from Figure 55 the regulation named "AA 4.72 (3)" will be expanded in a separate diagram.

By selecting and copying desired node (AA 4.72 (3)) from the main goal model or from the outline and copy/pasting it in a new empty GRL diagram, the situation is ready for expansion. We select expansion depths of 1, 2, and 3 levels, with the resulting diagrams shown in Figure 58, Figure 59 and Figure 60 respectively.

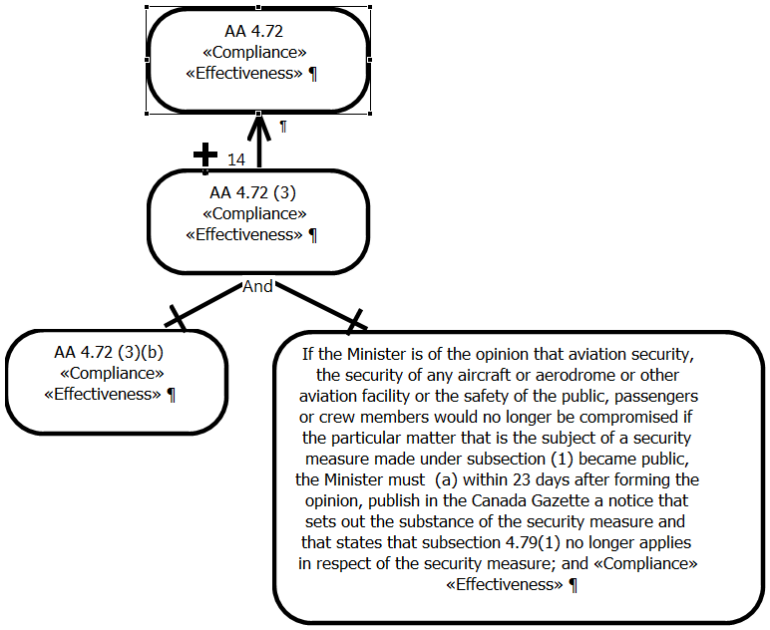


Figure 58 One Level Deep

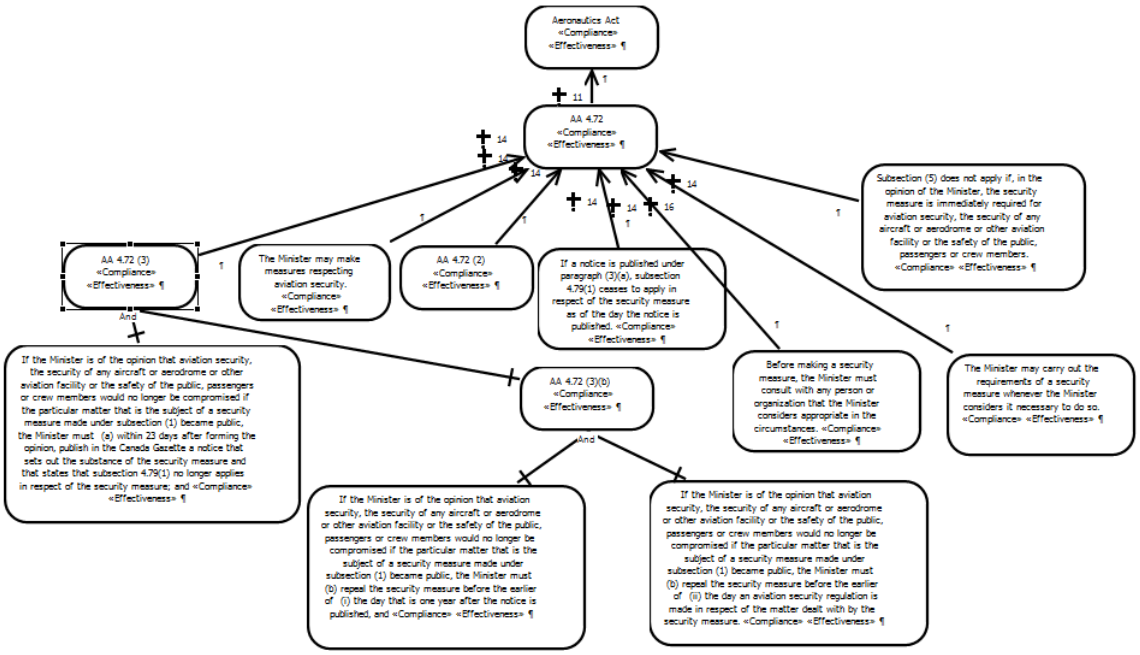


Figure 59 Two Levels Deep (Not Meant to Be Read)

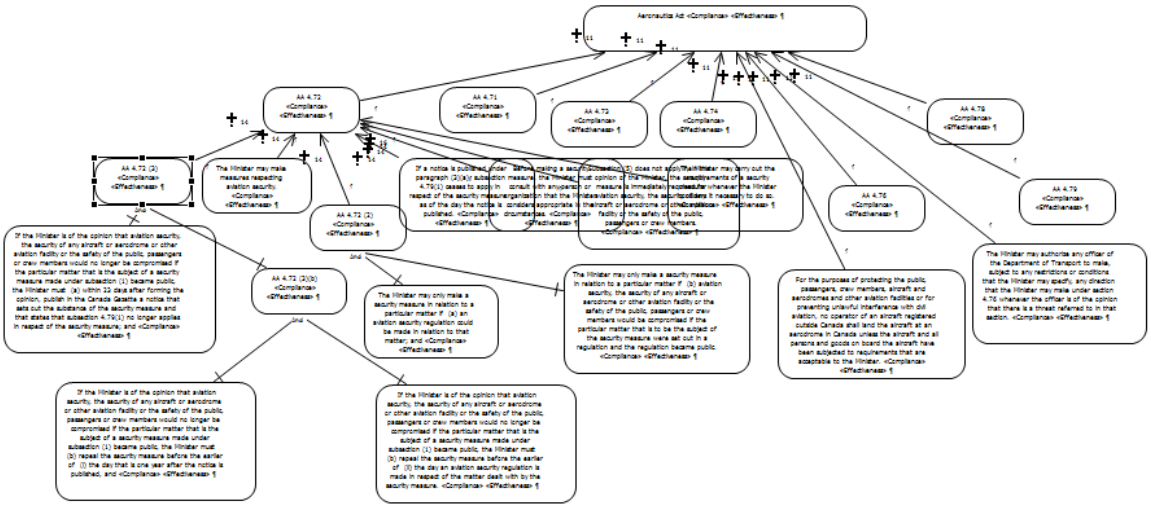


Figure 60 Three Levels Deep (Not Meant to Be Read)

There are other expansions provided as well. Those expansions are special case of general expansions. They are done in a way that is mostly useful for tree-structured goal models, such as the ones generated from regulations. In these expansions, only the links that are *incoming* will be expanded. Hence, this result in tree-shaped goal graphs. The number of levels of expansion for this type of expansion needs to be defined through a pop-up dialog box. Figure 61 and Figure 62 show examples of this type of expansion with one and three levels respectively. Note how simpler they are when compared to Figure 58 and Figure 60.

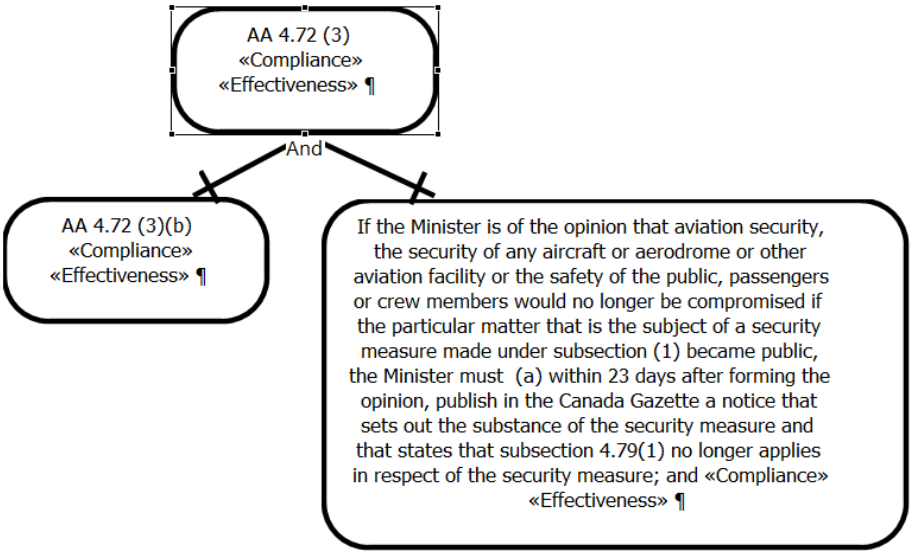


Figure 61 Expansion of Incoming Elements, One Level Deep

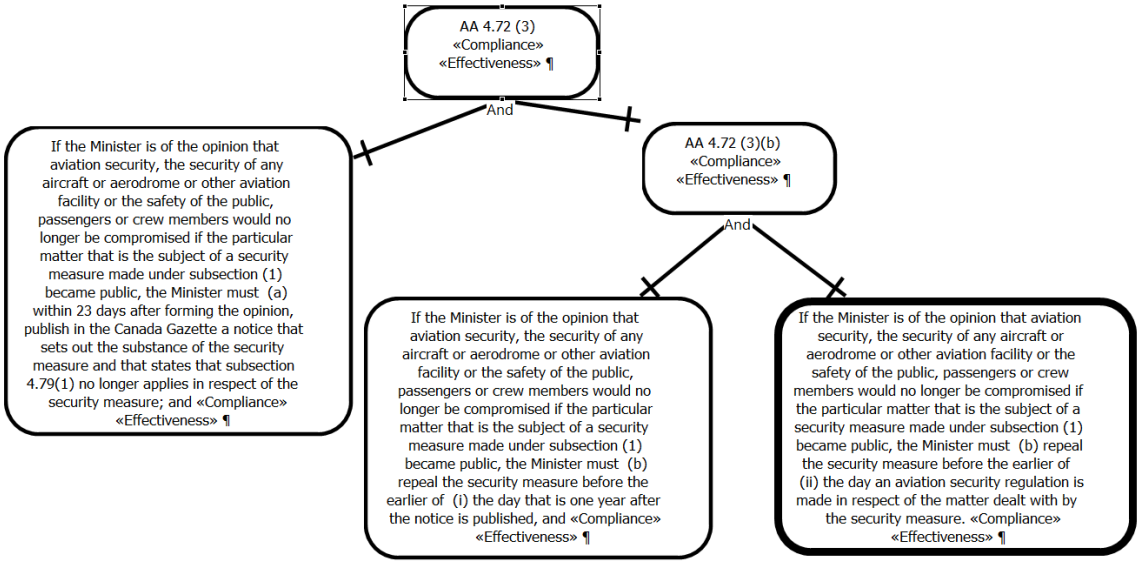


Figure 62 Expansion of Incoming Elements, Three Levels Deep

7.3.3 Evolution of Model (Improvement)

Following the first iteration, new information could be added and other modifications could improve automation. For example, the problem with regulations with similar IDs but different KPIs could be easily addressed by the regulator to simplify the generation of more precise models from the beginning (and it was addressed by them later).

7.4. Threats to Validity and Limitations

We have demonstrated that the approach and various tool-supported mechanisms developed in this thesis are applicable to regulations from three vastly different domains, with substantial (albeit not quantified) benefits in terms of efficiency and precision in model creation. There are however some threats and limitations in this work. These threats can be categorized in three different groups, as proposed by Perry et al. [46]:

- Construct validity: Specifies how well the case study answered the research question.
- Internal validity: Examines any bias and other confounding factors.
- External validity: Specifies how much of the results can be generalized.

7.4.1 Construct Validity

In all the case studies, we had access to regulations, but we never had access to complete information about them. In the third case study, because of confidentiality reasons, some fake (but realistic) information was provided alongside the regulations. In other case studies, since some of the essential information was missing, we had to create some information. In all cases, we used fake (and mostly equal) relative contribution weights. However, the non-availability of real information is not a major issue for validating the *automation* aspects of our work. Our focus was on the automated construction of models, and regulation structures (available for all case studies) were the most important aspect. The lack of realistic information is also as much a problem for our automated approach as it is for manual model construction approaches.

We have modelled legislations into regulation CSV files and KPI conversion CSV files. Therefore, it could be said that we took a specific approach (or a *default attitude*) to model them. If someone else tries to model similar legislations (even using and considering our guidelines from Section 4.3.2), they may model legislations in different ways, create CSV files slightly differently, and have different GRL regulation models at the end. Moreover, the thesis author has quite a bit of experience with URN and goal modelling. This fact may have unconsciously affected positively the creation of CSV files. To mitigate this issue, the first case study involved a colleague in the construction of the

CSV file. In addition, all three GRL models constructed here were reviewed by domain experts or peer-reviewed (in publications) in a satisfactory way. But again, bias in the construction of models would likely be similar for automated and manual approaches. Our approach would simply lead to "incorrect models" faster, giving more time to users to find model issues and to fix them, with less effort.

We have decided early on not to focus on Natural Language Processing techniques to help users detect decomposition relationships beyond those imposed by the structure of the document. In a sense, this limits our answer to the first research question (related to what can be automated). However, the exploration of such techniques is left to future work.

7.4.2 Internal Validity

This approach is based on the work of another student (Shamsaei) and tries to achieve automation for what she introduced in her manual approach. Thus, it was assumed that her approach toward modelling regulations with URN is correct, and we did not attempt to improve it beyond automation.

Moreover, different fields of information for regulation were considered during the development of this approach. Some of these fields are mandatory and some are optional, and we assumed this would be useful. This assumption about types of fields was based on our experience and that of fellow students in different domains. However, this assumption was not specifically and thoroughly tested in this thesis.

7.4.3 External Validity

Only three domains were used to validate this approach, mostly with one case study each (we have done additional work on other regulations related to aviation security with a similar level of success, but cannot report formally on this work here because of confidentiality reasons) covering parts of entire legislation. Generalization of the model construction approach to entire legislations or to other domains is hence premature at this time. However, these three very different domains and regulation structures helped build our confidence and we cannot foresee major generalization problems at this time.

Similarly, we have not systematically used the graph manipulation features on models from other domains, but we have ad hoc evidence they work on non-tree-like GRL models, and the next chapter covers additional unit testing of these features.

We have not studied the generalization of this approach to other goal modelling languages such i*, Nòmos, or KAOS. Although the other languages could likely benefit from a tabular representation for model construction, GRL offers unique features (such as indicators, metadata and strategies) that are not found in the other languages but that are key to our approach.

Finally, changes can happen in regulations. Although changes at that level are infrequent, some changes can introduce much work in updating CSV files and GRL models (especially when graphical views were added). However, regulation evolution is outside the scope of this thesis and is left for future work.

7.5. Summary

In this chapter, three applications of our proposed approach in different domains where introduced, explained and evaluated. The efficiency, (qualitative) performance and uniqueness of the thesis approach were demonstrated in these case studies. In addition, several important limitations and threats to validity were also discussed, together with mitigation approaches whenever available.

The next chapter provides the way we tested our implementation and presents a comparison between our work and three closely-related approaches already introduced in the literature review.

Chapter 8. Testing and Comparison

This chapter discusses in Section 8.1 the strategy used to test the tool implementation. Section 8.2 provides a comparison between three closely-related approaches from the literature review and our approach.

8.1. Implementation Testing

8.1.1 Testing of Goal Diagram Manipulation Features

We developed our implementation of the three categories of features in Chapter 6 using Java, and we used jUnit [31] for the automated testing of this part of our implementation. Since the features for manipulating goal diagrams were developed as *commands* (based on the Command design pattern) in jUCMNav, we needed to develop jUnit tests for each of these commands.

In the development environment of jUCMNav, there are already many test cases available for existing commands, and we simply added new command test cases to the current list. Our commands were tested from different points of view. For each command, *execution, undo* and *redo* methods were tested thoroughly. To be more specific, in these tests, a simple GRL model is created (through creation commands) and then a new GRL diagram is created and manipulated via our new manipulation commands. All these tasks got implemented in simple steps, and after each step an *assertion* (post-condition) was check to insure the accuracy of the transformed diagram. This GRL model is illustrated in Figure 63.

In term of coverage, we developed the jUnit test cases with different intentional elements as the starting node for expansion. We chose three different nodes: one from rood nodes (e.g., #1 in Figure 63), one from middle nodes (e.g., #4) and one from leaf nodes (e.g., #9).

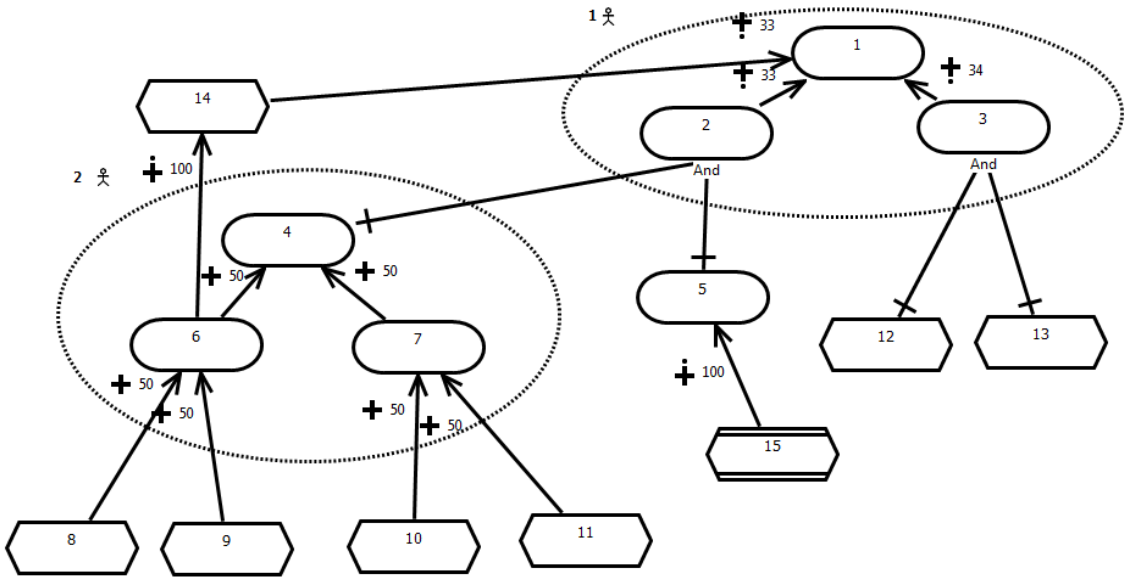


Figure 63 GRL Model for Testing Diagram Manipulation Commands

A list of all these automated test cases is provided below, and their code is found in the open-source code base of jUCMNav 5.4.0 [35]. In all of these test cases, goal #1 as head node, goal #4 as middle node, and task #9 as leaf node were chosen for expansion. Also, actor number 1 is used for testing in its respective test case. All of these tests have passed successfully.

- **testShowLinkedElementCommand**: One-level, two-level and three-level deep expansion commands are tested in this use case.
- **testShowLinkedElementAlternativeCommand**: This test case checks the advanced expansion command that requires defining the number of desired levels or that expands all levels.
- **testShowLinkedElementAlternativeSubNodesCommand:** This test case is used to test the advanced expansion command that requires defining the number of desired levels to show nodes that are connected through *incoming* links only.
- **testShowContainingElementCommand**: This test case tests the command that shows all the contained elements in an actor.
- **testShowContainingActorCommand**: This test case is used to test the command that shows all the containing actors of a node.

- **testShowLinkedElementCompleteCommand:** This test case is used to test the command that shows all existing and connecting nodes in a model.
- **testShowLinkedElementCompleteSubNodesCommand:** This test is used to test the command that shows all the existing and connecting nodes in a model that are connected through incoming links.

We also developed additional jUnit tests that used combinations of commands, with successful results.

8.1.2 Testing of the Automatic Generation of Goal Models

For this part of our implementation, which targets the two import mechanisms (regulations, and KPI conversions), we used a *testing by description* approach rather than a fully automated testing approach (which would have required too much effort). This approach focuses on the input domain of the software under study and tests the software in terms of expected outputs (results). Moreover, multiple classes of inputs are introduced. Selected inputs from the same class must provide similar results.

We tested the functionalities of our import mechanisms by selecting proper values for each parameter. Also, we needed to select proper environment variables to be able to conduct correct testing. We picked and tried different combinations of types and formats of inputs. Several issues were debugged along the way. For the selection space we covered in our testing, we no longer find any problem with our implementation. This was further validated on the HIPPA case study, done last with the software as is.

We created four groups of CSV files to test different aspects of our import mechanism. Different combinations of inputs got created and tested. In each groups, we tried to fill data fields with different types of data, including *null* values. We tried to remove the optional data fields and to evaluate the results of the generation. We even tried to evaluate the boundary of data fields wherever they were available. More information about the resulting 38 test cases for this part of our implementation is provided in Appendix B.

8.2. Comparison with Related Work

In Chapter 3, we have presented several existing approaches for generating goal models, and discussed them in terms of several criteria of interest in our context. Among all of the approaches we presented in this literature review, Shamsaei [56], Garu et al. [27] and Ingolfo et al. [32] are the ones closest to the approach developed in this thesis. In the following sections, we discuss and compare these approaches with ours in detail.

8.2.1 Context of Work

Shamsaei [56] focuses on creating goal models from regulation text, and on analyzing them in a business process improvement context. She proposed guidelines to parse regulation text and produce instances of a regulation metamodel. She also provided a mapping between regulation concepts from her metamodel and URN/GRL concepts.

Grau et al. [27] pay attention to information systems and their development (especially re-engineering). They propose an approach (PR*i*M) that uses *i** to model current processes and benefits from these models to evaluate and analyse patterns that can be used for the generation of new systems.

Ingolfo et al. [32] focus on goal-oriented compliance to regulations (with Nòmos) and how the quality of software requirements can be affected by it. Moreover, they discuss how conditions that exist in regulations can make life harder for software engineers, especially when faced with multiple alternatives.

As for the above, the application domain of our work is mainly related to legal compliance. We try to model regulations with goal models. We proposed an approach to have a quick and incremental development of GRL models based on an intuitive and easy-to-manipulate tabular presentation of regulations and their attributes and relationships.

8.2.2 Coverage of Criteria

As our work is based on Shamsaei's approach [56], it obviously is the most similar one among all related work. She generates goal models in GRL from regulations text and ad-

dresses different types of relations links and intentional elements while parsing regulations. She tags intentional elements in models to create a family of goal models that handles classes of regulations. She also addresses actors, indicators and conditions in her models. Although she provides good tools for the analysis of models, their construction remains largely a manual effort based on guidelines (and jUCMNav-based support for stereotyping elements). So, there is no automation provided for the *generation* of goal models.

Grau et al. [27] create *i** models that address alternatives in requirements engineering to comply with regulations. They provide a framework that can be developed iteratively and develop tool support for it. To use their approach, they support users with solid guidelines for all its steps. Their tool helps with the generation of *i** goal models systematically. However, it does not show goal diagrams in a traditional way, and rely on a tree-like structure. They make use of actors, intentional elements, and relationship links in their generated models. Also, their tool supports exploring models and provides traceability and rationale for model elements.

Ingolfo et al. [32] developed a tool to support their approach. The compliance goal models created are supposed to address the variability of alternatives. They have automatic reasoning approach implemented in their framework, but little support for the creation of models.

In our work, we developed an approach and extended an Eclipse-based plugin. Using our tool, we can generate goal models automatically from a tabular representation of regulations and their attributes, and dynamically explore model views for further analysis. When generating goal models, we use concepts of intentional elements, indicators and relationship links, but we have no automatic generation of actors. However, actors can be added after goal models are generated and features for manipulating them are developed. We support the concept of conditions in our goal models. Model elements in generated goal models can be tagged with metadata to carry necessary information helpful for users during analysis (for example alternative language, IDs, classes, etc.). Finally, since, we have multiple fields of information on regulations stored in our goal models, we can trace back and forth between regulations and their respective goals or source documents.

In Table 4, we summarize the comparison between our approach and closely-related work from the literature review.

Table 4 Comparison Between Our Approach and Closely-Related Work

Article	Automation	Decomposition	Contribution	Indicator	Actors	Traceability	Metadata	Diagram	Evolution	Alternative Language	Condition	Class (Stereotype)	Tool	Extendibility	Modelling Language
[27]	Yes	Yes	Yes	No	Yes	Yes	+/-	Yes	+/-	No	Yes	No	Yes	No	*i**
[56]	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	GRL
[32]	+/-	Yes	Yes	No	Yes	No	No	Yes	No	No	Yes	Yes	Yes	No	Nòmos
Thesis	Yes	Yes	Yes	Yes	+/-	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	GRL

8.3. Summary

We compared our work in this chapter with three closely-related approaches from the literature review. For the purpose expressed by the selected comparison criteria, our construction approach for goal-based regulation models scores better than the other approaches, and could be improved by further support for actors. In addition, the models generated benefit from a standardized language (GRL) with a mature analysis tool (jUCMNav).

In the next chapter, we conclude this thesis and provide future work items.

Chapter 9. Conclusions

This chapter summarizes contribution of the thesis in view of the research questions. It then introduces future work items of interest for further improving this approach.

9.1. Contributions

The research questions that we have addressed in this thesis were introduced in Section 1.2:

1. To what extent can we automate the generation of goal models from regulations?
2. Does this automation lead to models that are faster to produce and of higher quality?
3. How can the proposed approach help the regulatory compliance process?

In this thesis, a new approach for automatically constructing goal models from textual but structured regulations is introduced. This approach, illustrated in Figure 4, takes advantage of a tabular representation (in essence, a DSL with a CSV concrete syntax) of regulation concepts (defined in the metamodel of Figure 5), to be produced manually by compliance officers, regulators or lawyers according to new construction guidelines. As seen in the third case study, such information can also be partially generated from exiting regulation databases. Moreover, regulations can have conversion groups of evaluation values specific to their domain, and these groups are defined using a tabular presentation and stored in CSV files. These two CSV files are imported into jUCMNav to generate a GRL model where indicators are linked to their proper conversion group and where one diagram (will all elements and links) is generated. Such model is ready to be evaluated (e.g., for compliance analysis) through GRL strategies. In terms of validation, three case

studies from three different domains were used in order to cover different types of regulations.

The above **main contribution** of the thesis helps answer the first research question related to automation. As shown in Table 5, which recalls the concepts of the regulation metamodel needed to construct GRL models automatically, many concepts can be extracted from regulations, inspection questionnaires (or similar sources of indicators), and often existing databases. However, we demonstrated that three aspects of regulations (decomposition relationships, stereotypes for classes, and conditions) currently require manual parsing of regulation text. However, the latter *may be* parsed automatically in the future through techniques such as Natural Language Processing (NLP). There are also two concepts that simply do not exist in regulations and questionnaires: relative importance levels (for goal/KPI contributions) and KPI conversion mappings. These two cannot be automated without the involvement of experts, who can provide such values/mappings.

Table 5 Which Regulation Concepts Can Be Automatically Imported?

Class	Attribute	Automatable?
Regulation	Description	Yes (from regulation)
Regulation	LegislationID	Yes (from database)
Regulation	LegislationSection	Yes (from regulation)
Regulation	URL	Yes (from database)
KPI	Description	Yes (from questionnaires)
Contribution	Importance	No (requires experts)
Decomposition	Type	Maybe (perhaps through NLP)
Regulation	AlternativeDesc	Yes (from regulation)
KPI	AlternativeDesc	Yes (from questionnaires)
ConversionGroup	Name	No (requires experts)
Stereotype	Value	Maybe (perhaps through NLP)
Condition	Description	Maybe (perhaps through NLP)

To answer the second research questions related to models that are better and faster to produce, this thesis provides partial answers and contributions. We only have ad hoc evidence that the answer is yes, with informal feedback from model users, but there is supporting evidence in the form of a comparison with related work (as Shamsaei [56] observed, the creation of a goals/indicators for a regulation took her 20 minutes on average per regulation, whereas this takes less than a second with our tool), where our approach is the only one to cover the necessary concepts and to provide automation for the construction of regulation goal models. In addition, to support the iterative development and the evolution of models and regulations, the tabular format allows for optional fields and empty cells in some circumstances. Finally, a great accelerator for GRL view/diagram creation is provided by a **minor contribution** of this thesis composed of three families of expansion features developed for jUCMNav 5.4.0. Since regulation models are often considerably large, modellers can use these features to quickly create many new partial views (diagrams) form the main one, leading to more understandable model. Doing so without these features is a labour-intensive and error-prone task.

As for the third research question, this approach can support the regulatory compliance process in many ways (as evidenced by four publications extracted from this thesis [3][5][47][48]). One main contribution is that people without a modelling background will now be able to create goal models with familiar tools (e.g., Microsoft Excel), with an efficient syntax in that context. Also, as mentioned in the previous paragraph, this approach is suited for regulations under evolution, especially in a context where many old-fashion prescriptive regulations are being turned into outcome-based regulations [61]. The models generated by our approach can also be used as is for compliance analysis based on Shamsaei's techniques and algorithms [53][56] already implemented in jUCMNav, and her approach currently reflects the state of the art. We also already have a national regulator that is evaluating our tool-supported approach for adoption, and this is a good indicator of the pragmatism of our work.

9.2. Future Work

This work can be extended in many directions (from simpler ones to more complex ones).

1. Better support for actors in the input CSV format could easily be added (although this was not necessary for the three case studies we had).
2. Since Microsoft Excel is a likely candidate tool for manipulating the regulation CSV files, one could add Excel macros to validate the input and detect undesirable situations such as missing mandatory fields, the use of inexistent KPI conversion groups, etc. Such errors could hence be detected within Excel, without the need to import erroneous files jUCMNav.
3. Some additional graphical views (GRL diagrams) could be generated automatically at import time. For example, each intermediate node in the regulation model (with a parent and at least one child) could be turned into its own named diagram, with a one-level expansion.
4. Generalization of the work could be further explored by exploring more regulations, especially from other domains, and perhaps from different languages.
5. A usability study should be performed to assess quantitative benefits in terms of efficiency and error reduction. This could also lead to usability improvements in terms of the concrete CSV-based syntax, as well as to the set of guidelines we provided.
6. In order to better support the evolution of regulations, there is a need to preserve and evolve manually created diagrams/views, which would now be erased upon re-importing a modified CSV file. Being able to describe a diagram as a dynamic query on the model might be an interesting solution to explore.
7. The import of GRL models from a tabular representation could be generalized to non-tree-like models outside of the regulation domain. Our metamodel would need to include support for dependency links and for absent attributes currently found in the URN metamodel. In addition, the CSV concrete syntax would need more flexibility for linking any pair of intentional element (right now, the struc-

ture is limited to trees and cannot handle general graphs). This might create other and unexpected challenges along the way.

8. Finally, as discussed in Table 5, NLP techniques could be used to populate regulation CSV files from regulations and other sources of information. In particular, there might be some potential there in automating the extraction of decomposition relationships, stereotypes for classes, and conditions, which is currently done manually in our approach.

References

[1] Amyot, D. and Mussbacher, G.: "User Requirements Notation: The First Ten Years, The Next Ten Years". *Journal of Software*, Vol. 6, No. 5, Academy Publisher, May 2011, 747–768.

[2] Amyot, D., Ghanavati, S., Horkoff, J., Mussbacher, G., Peyton, L., and Yu, E.: "Evaluating Goal Models within the Goal-oriented Requirement Language". *International Journal of Intelligent Systems (IJIS)*, Vol. 25, Issue 8, August 2010, 841–877.

[3] Amyot, D., Rashidi-Tabrizi, R., Mussbacher, G., Kealey, J., Tremblay, E., and Horkoff, J.: "Improved GRL Modeling and Analysis with jUCMNav 5". *6th International i* Workshop (iStar 2013)*, Valencia, Spain, June 2013. CEUR-WS, Vol-978, 137–139.

[4] Amyot, D.: "Introduction to the User Requirements Notation: Learning by Example". *Computer Networks*, Volume 42, Issue 3, 285–301, June 2003.

[5] Badreddin, O., Mussbacher, G., Amyot, D., Behnam, S.A., Rashidi-Tabrizi, R., Braun, E., Alhaj, M., and Richards, G.: "Regulation-Based Dimensional Modeling for Regulatory Intelligence". *6th Int. Workshop on Requirements Engineering and Law (RELAW 2013)*, Rio de Janeiro, Brazil, July 2013. IEEE CS, 1–10.

[6] Boer, A., Hoekstra, R., and Winkels, R.: "Metalex: Legislation in XML". *Fifteenth Annual Conference on Legal Knowledge and Information Systems* (Jurix), 2002. OIS Press, 1–10.

[7] Brachman, R. and Levesque, H.: *Knowledge representation and reasoning*. Morgan Kaufmann, 2004.

[8] Braun, E., Cartwright, N., Shamsaei, A., Behnam, S.A., Richards, G., Mussbacher, G., Alhaj, M., and Tawhid, R.: "Drafting and Modeling of Regulations: Is It Being Done Backwards?". *5th International Workshop on Requirements Engineering and Law (RELAW 2012)*, Chicago, Illinois, USA, September 2012. IEEE CS, 1–6.

[9] Breaux, T.D., Vail, M.W., and Antón, A.I.: "Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations". *14th IEEE Int. Conference on Requirements Engineering (RE'06)*, 2006. IEEE CS, 49–58.

[10] Breaux, T.D. and Lotrionte, C.B.: "Towards a privacy management framework for distributed cybersecurity in the new data ecology", *IEEE International Conference on Technologies for Homeland Security (HST),* 2011. IEEE CS, 15–17.

[11] Breaux, T.D., and Gordon D.G.: *Regulatory requirements as open systems: structures, patterns and metrics for the design of formal requirements specifications*. Technical report, CMU-ISR-11-100, Carnegie-Mellon University, USA, 2011.

[12] Breaux, T.D., and Baumer, D.L.: "Legally "reasonable" security requirements: A 10-year FTC retrospective". *Computers & Security*, vol. 30, no. 4, 2011, 178–193.

[13] Breaux, T.D., Antón, A.I., Boucher, K., and Dorfman, M.: "Legal requirements, compliance and practice: an industry case study in accessibility". *16th IEEE Int. Conference on Requirements Engineering (RE'08)*, 2008. IEEE CS, 43–52.

[14] Chung, L., Nixon, B.A., Yu, E., and Mylopoulos J.: *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publisher, 2000.

[15] Elahi, G. and Yu, E.: "Modeling and analysis of security trade-offs–A goal oriented approach". *Data & Knowledge Engineering*, 68 (7), 2009, 579–598.

[16] Ellis-Braithwaite, R., Lock, R., Dawson, R., and Haque, B.: "Modelling the Strategic Alignment of Software Requirements using Goal Graphs". *International Conference on Software Engineering Advances, 2012*. IARIA, arXiv:1211.6258.

[17] Faily, S. and Fléchais, I.: "Towards tool-support for usable secure requirements engineering with CAIRIS". *International Journal of Secure Software Engineering (IJSSE)*, Vol. 1, No. 3, 2010, 56–70.

[18] Faily, S.: "Bridging User-Centered Design and Requirements Engineering with GRL and Persona Cases". *5th International i* Workshop (iStar 2011)*, Trento, Italy, August 2011. CEUR-WS Vol-766, 114–119.

[19] Franch, X., Grau, G., Mayol, E., Quer, C., Ayala, C., Cares, C., Navarrete, F., Haya, M., and Botella, P.: "Systematic construction of *i** strategic dependency models for socio-technical systems". *Int. J. of Software Engineering and Knowledge Engineering*, Vol. 17, No. 1, 2007, 79–106.

[20] Ghanavati, S., Amyot D., and Peyton, L.: "Towards a Framework for Tracking Legal Compliance in Healthcare". *19th Int. Conf. on Advanced Information Systems Engineering (CAiSE'07)*. Trondheim, Norway, June 2007. LNCS 4495, Springer, 218–232.

[21] Ghanavati, S., Amyot, D., and Peyton, L.: "A Systematic Review of Goal-oriented Requirements Management Frameworks for Business Process Compliance". *4th Int. Workshop on Requirements Engineering and Law (RELAW)*, Trento, Italy, August 2011. IEEE CS, 25–34.

[22] Ghanavati, S., Amyot, D., and Peyton, L.: "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, 133–142.

[23] Ghanavati, S., Siena, A., Amyot, D., Perini, A., Peyton, L., and Susi, A.: "Integrating Business Strategies with Requirement Models of Legal Compliance". *Int. J. Electronic Business*, 8(3), Inderscience Publishers, 2010, 260–280.

[24] Ghanavati, S.: *Legal-URN Framework for Legal Compliance of Business Processes*. Ph.D. thesis, School of Electrical Engineering and Computer Science, University of Ottawa, Canada, 2013. http://www.ruor.uottawa.ca/en/handle/10393/24028 (accessed May 2013)

[25] Gordon, D.G. and Breaux, T.D.: Assessing Regulatory Change through Legal Requirements Coverage Modeling. *21st IEEE Int. Conference on Requirements Engineering (RE'13)*, 2013. IEEE CS (to appear).

[26] Grau, G., Franch, X., and Avila, S.: "J-PRiM: A Java tool for a process reengineering *i** methodology". *14th IEEE Int. Conference on Requirements Engineering (RE'06)*, 2006. IEEE CS, 359–360.

[27] Grau, G., Franch, X., and Maiden, N.A.M.: "PR*i*M: An i∗-based process reengineering method for information systems specification". *Information and Software Technology*, Vol. 50. No. 1, 2008, 76–100.

[28] Hevner, A.R., March, A.T., Park, J., and Ram, S.: "Design science in information systems research". *MIS Quarterly*, Vol. 28, No.1, March 2004, 75–105.

[29] Hilts, A., and Yu, E.: "Design and evaluation of the goal-oriented design knowledge library framework". *Proceedings of the 2012 iConference*, 2012. ACM, 384–391.

[30] http://opencsv.sourceforge.net/ (accessed September 2012)

[31] http://junit.org/ (accessed May 2013)

[32] Ingolfo, S., Siena, A., Jureta, I., Susi, A., Perini, A., and Mylopoulos, J.: "Choosing compliance solutions through stakeholder preferences". *Requirements Engineering: Foundation for Software Quality*, 2013. LNCS 7830, Springer, 206–220.

[33] Ishikawa, F., Inoue, R., and Honiden, S.: "Modeling, analyzing and weaving legal interpretations in goal-oriented requirements engineering". *Second International Workshop on Requirements Engineering and Law (RELAW)*, 2009. IEEE CS, 39–44.

[34] ITU-T, Recommendation Z.151 (10/12): *User Requirements Notation (URN) – Language Definition*, Geneva, Switzerland, October 2012.

[35] jUCMNav, Version 5.4.0, University of Ottawa, June 2013. http://softwareengineering.ca/jucmnav/

[36] jUCMNav, Version 5.4.0 Metamodel, University of Ottawa, June 2013. http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/URNMetaModel

[37] Kitchenham, B.: *Procedures for performing systematic reviews*. Technical Report, Keele University and NICTA, Staffordshire, UK, 2004.

[38] Kiyavitskaya, N., Zeni, N., Mich, L., Cordy, J.R., and Mylopoulos, J.: "Text mining through semi-automatic semantic annotation". *Practical Aspects of Knowledge Management*, 2006. LNCS 4333, Springer, 143–154.

[39] Kiyavitskaya, N., Zeni, N., Breaux, T.D., Antón, A.I., Cordy, J.R., Mich, L., and Mylopoulos, J.: "Automating the extraction of rights and obligations for regulato-

ry compliance". *Conceptual Modeling-ER 2008*, 2008. LNCS 5231, Springer, 154–168.

[40] Lockerbie, J., Bush, D., Maiden, N., Blom, H., and Everdij, M.: "Using *i** Modelling as a Bridge between Air Traffic Management Operational Concepts and Agent-based Simulation Analysis". *18th IEEE Int. Requirements Engineering Conference (RE'10),* 2010. IEEE CS, 351–356.

[41] Maxwell, J.C. and Antón, A.I.: "Developing production rule models to aid in acquiring requirements from legal texts". *17th IEEE Int. Requirements Engineering Conference (RE'09)*, 2009. IEEE CS, 101–110.

[42] Maxwell, J.C. and Antón, A.I.: "The production rule framework: developing a canonical set of software requirements for compliance with law". *Proceedings of the 1st ACM International Health Informatics Symposium*, 2010. ACM, 629–636.

[43] Munro, S., Liaskos, S., and Aranda, J.: "The mysteries of goal decomposition". *5th International i* Workshop (iStar 2011)*, Trento, Italy, August 2011. CEUR-WS Vol-766, 49–54.

[44] Mussbacher, G., Amyot, D., and Heymans, P.: "Eight Deadly Sins of GRL". *5th International i* Workshop (iStar 2011)*, Trento, Italy, August 2011. CEUR-WS Vol-766, 2–7.

[45] Oliveira, A. P. A., do Prado Leite, J. C. S., Cysneiros, L. M., & Lucena, C. J. P.: "*i** Diagnoses: A Quality Process for Building *i** Models". *Proceeding of the Forum at the CAISE'08 Conference*, 2008, CEUR-WS, Vol-344, 9–12.

[46] Perry, D.E., Porter, A.A., and Votta, L.G.: "Empirical studies of software engineering: a roadmap". *Proc. Conference on The future of Software Engineering*, 2000. ACM, 345–355.

[47] Rashidi-Tabrizi, R., Mussbacher, G., and Amyot, D.: "Legal Requirements Analysis and Modeling with the Measured Compliance Profile for the Goal-oriented Requirement Notation". *6th Int. Workshop on Requirements Engineering and Law (RELAW 2013)*, Rio de Janeiro, Brazil, July 2013. IEEE CS, 53–56.

[48] Rashidi-Tabrizi, R., Mussbacher, G., and Amyot, D.: "Transforming Regulations into Performance Models in the Context of Reasoning for Outcome-Based Compliance". *6th Int. Workshop on Requirements Engineering and Law (RELAW 2013)*, Rio de Janeiro, Brazil, July 2013. IEEE CS, 34–43.

[49] Rifaut, A.: "Compliance management with measurement frameworks". *Fourth Int. Workshop on Requirements Engineering and Law (RELAW)*, 2011. IEEE CS, 15–24.

[50] Rifaut, A. and Ghanavati, S.: "Measurement-oriented comparison of multiple regulations with GRL". *Fifth Int. Workshop on Requirements Engineering and Law (RELAW),* 2012. IEEE CS, 7–16.

[51] Roberson, S. and Robertson, J.: *Volere: Requirements Specification Template*. The Atlantic Systems Guild, 2012.

[52] Roy, J.-F.: *Requirement Engineering with URN: Integrating Goals and Scenarios*. Master of Computer Science thesis, School of Information Technology and Engineering, University of Ottawa, Canada, 2007.

[53] Shamsaei, A., Amyot, D., Pourshahid, A., Yu, E., Mussbacher, G., Tawhid, R., Braun, E., and Cartwright, N.: "An Approach to Specify and Analyze Goal Model Families". *System Analysis and Modeling: Theory and Practice, 7th SAM Workshop*, Innsbruck, Austria, October 2012. LNCS 7744, Springer, 34–52.

[54] Shamsaei, A., Pourshahid, A., and Amyot, D.: "A Systematic Review of Compliance Measurement Based on Goals and Indicators". *Third Int. Workshop on Governance, Risk and Compliance – Applications in Information Systems (GRCIS 2011)*, CAiSE 2011 Workshops, June 2011. LNBIP 83, Springer, 228–237.

[55] Shamsaei, A., Pourshahid, A., and Amyot, D.: "Business Process Compliance Tracking Using Key Performance Indicators". *6th International Workshop on Business Process Design (BPD 2010)*, Hoboken, USA, September 2010. BPM 2010 Workshops, LNBIP 66, Springer, 73–84.

[56] Shamsaei, A.: *Indicator-based Policy Compliance of Business Processes*. Ph.D. thesis, School of Electrical Engineering and Computer Science, University of Ottawa, Canada, 2012. http://hdl.handle.net/10393/23476 (accessed January 2013)

[57] Sherman, D.M.: "A Prolog model of the income tax act of Canada". *Proceedings of the 1st international conference on artificial intelligence and law*, 1987. ACM, 127–136.

[58] Siena, A.: *Engineering Law-Compliant Requirements. The Nòmos Framework*. Ph.D. thesis, DISI, University of Trento, Italy, March 2010. https://sites.google.com/site/albertosiena/ (accessed February 2013)

[59] Siena, A., Ingolfo, S., Susi, A., Jureta, I., Perini, A., and Mylopoulos, J.: "Requirements, intentions, goals and applicable norms". *Advances in Conceptual Modeling*, 2012. LNCS 7518, Springer, 195–200.

[60] Strohmaier, M., Horkoff, J., Yu, E., Aranda, J., and Easterbrook, S.: "Can Patterns improve *i** Modeling? Two Exploratory Studies". *Requirements Engineering: Foundation for Software Quality*, 2008. LNCS 5025, Springer 153–167.

[61] Tawhid, R., Alhaj, M., Mussbacher, G., Braun, E., Cartwright, N., Shamsaei, A., Amyot, D., Behnam, S.A., and Richards, G.: "Towards Outcome-Based Regulatory Compliance in Aviation Security". *20th IEEE Int. Requirements Engineering Conf. (RE'12)*, Chicago, USA, September 2012. IEEE CS, 267–272.

[62] van Gog, R. and van Engers, T.M.: "Modeling legislation using natural language processing". *Systems, Man, and Cybernetics, 2001 IEEE International Conference on IEEE.* vol. 1. IEEE CS, 561–566.

[63] van Lamsweerde, A.: *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[64] Young, J.D., and Anton, A.I.: “A method for identifying software requirements based on policy commitments”. *18th IEEE International on Requirements Engineering Conference (RE’10)*, 2010. IEEE CS, 47–56.

[65] Yu, E.S.-K.: *Modelling strategic relationships for process reengineering*. Ph.D. thesis, Dept. of Computer Science, University of Toronto, Canada, 1995.

Appendix A: Parts of Implementation Code

This appendix presents important and illustrative parts of the implementation of the new regulation import mechanism in jUCMNav.

A.1. Reading Information from the CSV File

```
public URNspec importURN(String filename,
Vector autolayoutDiagrams) throws
InvocationTargetException {
   FileInputStream grlFileStream = null;
   CSVFile legislationFileSimple = new CSVFile(filename);
   XMLFileSimple xmlFileSimple = new XMLFileSimple();

   // Reading and cleaning up file, see Appendix A.2
   legislationFileSimple.ReadFile();
   legislationFileSimple.deleteBlanks();
   ....

   return urn;
}
```

A.2. Reading Information with OpenCSV

```
public class CSVFile {
   private String fileName;
   private List <String[]> stringValues;
   ...
   // to read a csv file into a list of strings
   public void ReadFile() {
```

```
      try {
            CSVReader reader = new CSVReader( new FileReader(
                                                fileName ), ',' );
            stringValues = (List<String[]>) reader.readAll();
         } catch ( IOException e ) {
         e.printStackTrace();
      }
   }

   // to remove all the '\n's that are created by conversion of Excel
   // file to tab-delimited file
   public void deleteBlanks() {
      String tempStr;
      String [] stringArray;
      for ( int i = 1; i < stringValues.size(); i++ ) {
         stringArray = stringValues.get( i );
         for ( int j = 0; j < stringArray.length; j++ ) {
            tempStr = stringArray[ j ].replaceAll( "\n", "");
                        stringArray[ j ] = tempStr;
            }
         }
      }
}
```

A.3. Necessary Data Structures to Store Information

```
   public class XMLFileSimple {
      private ArrayList <String> LegislationID;
      private ArrayList <String> LegislationSection;
      // a list of legislation with all the elements being labeled with
      // relevant name
      private ArrayList <String> RefinedLegislationSection;
      // ENGLISH-DESCRIPTION values
      private ArrayList <String> IntentionalElement;
      private ArrayList <String> RedundantIntentionalElement;
      private ArrayList <String> AltName; // FRENCH_DESCRIPTION values
      private ArrayList <String> hyperlink; //URL values
```

```
private ArrayList <String> AltDescription; // related to grl file
private ArrayList <String> Importance;
private ArrayList <ArrayList<String>> ImportanceRow;
private ArrayList <String> ImportanceRange;
private ArrayList <String> Decomposition;
private ArrayList <String> Kpi;
private ArrayList <ArrayList <String>> MultipleKpi;
private ArrayList <String> NoRedundantMultipleKpi;
private ArrayList <String> RedundantMultipleKpi;
private ArrayList <String> KpiWeight;
private ArrayList <ArrayList<String>> MultipleKpiWeight;
private ArrayList <String> AltKpi;
private ArrayList <ArrayList <String>> MultipleAltKpi;
private ArrayList <String> NoRedundantMultipleAltKpi;
private ArrayList <String> RedundantMultipleAltKpi;
private ArrayList <ArrayList<String>> GoalKpiWeight;
private ArrayList <ArrayList<Integer>> KpiContributionValueList;
private ArrayList <String> Stereotype;
private ArrayList <ArrayList<String>> GoalStereoType;
private ArrayList <ArrayList<String>> MultipleStereoType;
// to consider all the elements AND, except for OR private Ar-
rayList <String> RefinedDecomposition;
// for grl file
private ArrayList <ElementDefinition> ElementDefinitionList;
// for KPIs in grl file
private ArrayList <ElementDefinition> KpiElementDefinitionList;
private ArrayList <ContributionAttribute> KpiContributionLinkDef-
initionList; // for KPIs in grl file
private ArrayList <DecompoitionAttribute> DecompositionLinkDefi-
nitionList; // for grl file
private ArrayList <ContributionAttribute> ContributionLinkDefini-
tionList; // for grl file
private ArrayList <RelationNode> RelationList; // to keep rela-
tions betweeen goals
private ArrayList <String> IDList; // list of ids of element def-
initions of grl file elements
private ArrayList <String> LinkIDList; // list of ids oflink def-
inition of grl file elements
```

```
    private ArrayList <Integer> ContributionValueList;

    private boolean noKpi = true, noAltKpi = true, kpiStereo
    typeExists = false, goalStereotypeExists = false,
    noImportanceRange = true;
    private int numberofIntentionalElement, endOfGoalsIndex;
    private final static int metadataArraySize = 5;
    private static int columnSize;
    private static int currentColumn = 0;
    ...
}
```

A.4. Some Illustrative Setter & Getter Methods

```
public void setLegislationID( List <String[]> list ) {
    String [] row;
    columnSize = list.get(0).length;
    currentColumn = 0;
    for (int i = 0; i < list.size(); i++)
       // finding the last row of goals in the CSV file
          if (list.get(i) [0].equals("KpiStereoTypes")) {
             endOfGoalsIndex = i;
             break;
          } else
                endOfGoalsIndex = list.size();

    LegislationID = new ArrayList<String>();
    if (currentColumn < columnSize)
          for (int i = 0; i < endOfGoalsIndex; i++) {
                row = list.get(i);
                LegislationID.add(row[currentColumn]);
          }

    currentColumn++;
    LegislationID.remove(0);
}
```

```
public ArrayList <String> getLegislationID() {
   return LegislationID;
}

public void setLegislationSection( List <String[]> list ) {
   String [] row;
   LegislationSection = new ArrayList<String>();
   if (currentColumn < columnSize)
         for ( int i = 0; i < endOfGoalsIndex; i++ ) {
               row = list.get( i );
               LegislationSection.add( row[ currentColumn ] );
         }

        LegislationSection.remove( 0 );
        currentColumn++;
}

private void setRefinedLegislationSection() {
   // make new copy of list of LegislationSection into
   // RefinedLegislationSection
   RefinedLegislationSection = new ArrayList <String>
                                              (LegislationSection);
   for (int i = 0; i < LegislationSection.size(); i++)
      // Creating RefinedLegislationList
         if (LegislationSection.get(i).equals(""))
               RefinedLegislationSection.set(i,
                                       IntentionalElement.get(i));
}

public ArrayList <String> getLegislationSection() {
   return LegislationSection;
}
```

```
public ArrayList <String> getRefinedLegislationSection() {
   return RefinedLegislationSection;
}

public void setIntentionalElement( List <String[]> list ) {
   String [] row;
   IntentionalElement = new ArrayList<String>();
   if (currentColumn < columnSize)
         for ( int i = 0; i < endOfGoalsIndex; i++ ) {
               row = list.get( i );
               IntentionalElement.add( row[ currentColumn ] );
         }

   currentColumn++;
   IntentionalElement.remove( 0 );
   setRefinedLegislationSection();
   // Taking care of redundant regulation I
   findRedundantIntentionalelement();
   // Taking care of redundant regulation II
   refineIntenionalElement();
   numberofIntentionalElement = IntentionalElement.size();
}

public ArrayList <String> getIntentionalElement() {
   return IntentionalElement;
}
```

A.5. Recognizing the Structure of the Document

```
// This function starts making the relation list by considering the
// LegislationSection, Importance and Decomposition list
void makeRelationList() {
   String tempFather = "";
   RelationNode rNode;
   RelationList = new ArrayList <RelationNode>();
```

```
for (int i = 0; i < LegislationSection.size(); i++) {
   // Creating an empty RelationList with the size of number of
   // rows in CSV file minus 1
     rNode = new RelationNode();
     rNode.setName(IntentionalElement.get(i));
     rNode.setFather("");
     RelationList.add(rNode);
}

RelationList.get(0).setName(LegislationSection.get( 0 ));
// Setting first node's father in RelationList
RelationList.get(0).setFather("");
for (int i = 1; i < RefinedLegislationSection.size(); i++) {
     // Continuing with making relation list
      tempFather = RefinedLegislationSection.get(i);
      for (int j=i+1; j < RefinedLegislationSection.size(); j++)
      if   (RefinedLegislationSection.get(j).contains(tempFather))
      {
            RelationList.get(j).
               setName(RefinedLegislationSection.get(j));
           RelationList.get(j).setFather(tempFather);
      }
}

for (int i = 1; i < RelationList.size(); i++)
// setting the name of parent of the immediate nodes after the
// top-level node (that do not have any name yet) to the name of
// the top-level node
      if (RelationList.get(i).getFather().equals("")) {
            RelationList.get(i).setFather(RelationList.get(0).
               getName());
            RelationList.get(i).
               setName(RefinedLegislationSection.get(i));
      }
}
```

A.6. Reading and Processing KPI Conversion Groups

```
// Reading and cleaning up file
qualitativeMappingCSVFile.ReadFile();
qualitativeMappingCSVFile.deleteBlanks();

// Creating Qualitative Mapping list
qualitativeMappingSimple.setRealWorldLabelList (qualitative-
MappingCSVFile.getStringValues());
qualitativeMappingSimple.setEvaluationList (qualitative-
MappingCSVFile.getStringValues());
qualitativeMappingSimple.setExceedsList (qualitative-
MappingCSVFile.getStringValues());
qualitativeMappingSimple.setKPICoversionNames();
```

A.7. Assigning KPI Conversion Groups Using Synchronized Threads

```
Display.getDefault().syncExec(new Runnable() {
@Override
public void run() {
   int j = 1;
   for (int counter = 0; counter < numberOfBorder; counter++) {
      // for the number of borders
      CreateKPIConversionCommand cmd1 = new CreateKPIConversionCom-
      mand (urnSpec, QualitativeMappings.class);
      cmd1.execute(); // creating mapping groups
      qualitativeMaps = (QualitativeMappings)
      cmd1.getKPIConversion();
      for (int i = j; i < numberOfMappings; i++) {
         if (!qualitativeMappingSimple.getRealWorldLabelList().
           get(i).contains("RealWorldLabel")) {
               qualitativeMaps.setName (
                  qualitativeMappingSimple.
                  getKPICoversionNames().get(counter));
```

```
                CreateQualitativeMappingCommand cmd2 = new
                   CreateQualitativeMappingCommand(urnSpec,
                                                   qualitativeMaps);
                cmd2.execute(); // creating maps
                qualitativeMap = cmd2.getMapping();
                qualitativeMap.setRealWorldLabel (
                   qualitativeMappingSimple.getRealWorldLabelList().
                  get(i));
                qualitativeMap.setEvaluation (Integer.
                   arseInt((qualitativeMappingSimple.
                   getEvaluationList().get(i))));
                qualitativeMap.setExceeds
                  (qualitativeMappingSimple.getExceedsList().get(i));
                j++;
               }
               else {
                j++;
                    break;
                }
                } // end of second for
         } // end of first for
   } // end of run
});
```

A.8. Calculating Contribution Values from Importance Values

```
// This method calculates values for contribution in GRL graph using
// the introduced values in Importance Column.
public void calculateContributionValueList() {
        String father;
        ArrayList <Integer> siblingsList;
        int weightSum = 0, contributionSum = 0,
            contributionValue = 0;

        ContributionValueList = new ArrayList<Integer>();
```

```
for ( int i = 0; i < Importance.size(); i++ )
  //Initialize ContributionValueList to 0 for all elements
    ContributionValueList.add( 0 );

System.out.println("size of ContributionValueList is :
                          "+ContributionValueList.size());
for (int i = 1; i < Importance.size(); i++) {
    // now, we are calculating!!!
    if (!Importance.get(i).equals("") &&
                  ContributionValueList.get(i) == 0) {
        contributionSum = 0;
        contributionValue = 0;
        weightSum = 0;
        father = RelationList.get(i).getFather();
        siblingsList = new ArrayList<Integer>();
        siblingsList.add(i);
        weightSum = Integer.parseInt(Importance.get( i ));
        for (int j = i + 1; j < Importance.size(); j++)
          // Finding other siblings if there is any
            if (RelationList.get(j).getFather().
                  equals(father))
             // Checking to find rows with same father
                if (!Importance.get(j).equals("")) {
                   // Check for being a contribution
                    siblingsList.add(j);
                    // keeping its index
                    weightSum = weightSum +
                              Integer.parseInt(
                                    Importance.get(  j  )  );
                              // adding its weight
                }
```

```
                for (int k = 0; k < siblingsList.size(); k++) {
                     contributionValue = ( Integer.parseInt(
                            Importance.get(siblingsList.get(k)) )
                            * 100 ) / weightSum;
                     contributionSum =  contributionSum +
                                        contributionValue;
                     if (k < (siblingsList.size() - 1))
                         ContributionValueList.set(
                             siblingsList.get(k), contributionValue);
                     if (k == (siblingsList.size() - 1)) {
                         if (contributionSum != 100)
                             ContributionValueList.set(
                                  siblingsList.get(k),
                                  contributionValue +
                                  (100 - contributionSum));
                         else
                             ContributionValueList.set(
                             siblingsList.get(k), contributionValue);
                     }
                 }
             }
        }
}
```

Appendix B: Conversion Test Cases

This appendix explains the content of the test cases for the model generation capabilities. We divided our test cases into four different groups. In each group, we targeted the successful generation of models from CSV files, although there are cases of failure presented too. For each test case, a short explanation is provided to explain its rationale. For test cases with successful results, the generated models are also provided. The test cases are available online at http://www.eecs.uottawa.ca/~damyot/pub/Rashidi/AppendixB.zip.

In the first group (10 test cases), we use CSV files with an incomplete set of data. In some cases, *null* values are provided. Also, we test our implementation against small models (to evaluate different sizes of regulations). In this group, KPI stereotypes, KPIs contributing to multiple regulations, and KPI conversion groups are not tested.

In the second group of the developed case studies (7 test cases), we consider KPI conversion groups and create different situations to test the tolerance factor of our implementation. Moreover, having more complete models compared to the first group, we evaluate model generation with different data fields present or removed from CSV files.

In the third group (10 test cases), we evaluate the boundaries between data fields wherever they exist. We move from simple CSV files to a very complicated one and evaluate the effects of existence/absence of data fields and the boundary separating labels in CSV files. In one case, model generation was not possible. This group of test cases also evaluates the tolerance and evaluation factors.

Finally, the most complicated and complete CSV files are presented in the fourth group (11 test cases). We study the advanced features mostly related to KPIs. Moreover, we change (removed/present) one or two data fields in each CSV file compared to prior files. In this group, some test cases involve a CSV file for KPI conversion groups.

Altogether, these 38 test cases ensure a good coverage of the features of the conversion tool.