Pontifícia Universidade Católica
do Rio de Janeiro

Priscila Engiel

Eunomia (Εὐνομία):
A Requirement Engineering based Compliance Framework for Software Systems

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências - Informática

Advisor: Prof. Julio Cesar Sampaio do Prado Leite
Co-Advisor: Prof. John Mylopoulos

Rio de Janeiro
February 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Priscila Engiel

Eunomia (Εὐνομία):
A Requirement Engineering based Compliance Framework for Software Systems

Thesis presented to the Programa de Pós-Graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Ciências – Informática. Approved by the undersigned Examination Committee

Prof. Julio Cesar Sampaio do Prado Leite
Advisor
Departamento de Informática – PUC-Rio

Prof. John Mylopoulos
Co-Advisor
UNITN

Prof.Jaelson Freire Brelaz de Castro
UFPE

Prof[a]. Claudia Cappelli
UNIRIO

Prof. Sérgio Lifschitz
Departamento de Informática – PUC-Rio

Prof[a]Vera Maria Benjamim Werneck
UERJ

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Técnico Científico da PUC-Rio

Rio de Janeiro, February 7th, 2018

Priscila Engiel

The author graduated in information systems from UNIRIO (Federal University of the State of Rio de Janeiro) in 2009. She completed his master's degree in Information Systems from UNIRIO in 2012, researching how to provide transparency to business processes, improving the representation of process models so that they are understood by citizens. It develops research related to requirements engineering, processes, legal compliance and transparency. Participated in several congresses in the area of requirements engineering. She is a lecturer at CCE (Central Extension Coordination) at PUC-Rio teaching about requirements engineering and business process management.

Bibiliographic data

Engiel, Priscila

Eunomia (Εὐνομία): A Requirement Engineering Based Compliance Framework for Software Systems / Priscila Engiel; advisor: Julio Cesar Sampaio do Prado Leite ; co-advisor: John Mylopoulos. – 2018.

141 f. : il. color. ; 30 cm

Tese (doutorado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2018.

Inclui bibliografia

1. Informática – Teses. 2. Conformidade legal. 3. Requisitos legais. 4. Engenharia de requisitos. 5. Conformidade contínua. I. Leite, Julio Cesar Sampaio do Prado. II. Mylopoulos, John. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

To my mom, Maria Ethelvina Engiel, who always encouraged me in academic life and followed all my steps.

Acknowledgements

To my mother, Ethel, the person that encourages me a lot to follow my dreams, the one that taught me that nothing is impossible, the person that is on my side, even though on another plane. Thank you for teaching me how to face the challenges with your head held and with a smile on your face. However large and impossible they may appear, we will always be able to overcome them, and we will always learn from them.

To my father, Celso, for all the support, understanding and encouragement. Thank you for supporting my decisions, even often not agreeing with them. Thank you for always being there.

To my sister, Deborah, who has always served as an example and focus of admiration. Thank you for showing me that the world does not have to be perfect and that moments off light and madness are often more than necessary. Thank you for all the admiration and security you have in me. Thank you for understanding that I had to leave.

To my advisers, Julio and John, for the patience, attention, availability and discussions throughout this work. Thank you for teaching me to be less anxious, immediate and practical. To teach me that the doctorate is not a project and that research needs patience. Today I look back and see how much knowledge has been acquired along the way, not just academic knowledge but learning that I will take with me always.

To my friends, who had the patience to listen so often on topics that appeared to be from another world, because they understand the moments of absence, by the fans for my success, and the joy and pride in my small achievements. Especially to Renata Benoliel and Renata Farhi, my sisters from another mom.

To the friendships conquered during the doctorate, Joanna, Henrique, Roxanna, Evelin, Luca, Elias, Tiago and Elda. I'm sure you made this journey more pleasant and enjoyable. Thank you for the conversations, for sharing the dramas, for the encouragement, for making the year in Italy lighter and Trento a second home. Having you on my side experiencing the same conflicts was essential for this work.

To Andrea and Juliana for believing in me when I was not believing in myself, for not letting me give up, for the spores, for the reviews and especially for advice. You are examples of researchers that I will always take with me.

To CAPES and to PUC-Rio for funding this research and making this work possible.

Abstract

Engiel, Priscilla; Sampaio do Prado Leite, Julio Cesar (Advisor); Mylopoulos, John (Co-Advisor). **Eunomia (Εὐνομία): A Requirement Engineering based Compliance Framework for Software Systems**, Rio de Janeiro, 2018, 141p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Laws and regulation affect software development, as they frequently demand changes in software' requirements to protect individuals and businesses regarding security, privacy, governance, sustainability and more. Legal requirements can dictate new requirements or constrain existing ones. The problem of software compliance is how to ensure that the software complies with the norms that the legislation imposes. The problem is particularly challenging because it combines difficultsteps: 1)analyze legal documents, 2) extract requirements from those documents, 3) identify conflictingrequirements with those already implemented in software and4) ensure that software remains compliant even with the changes. Compliance is a continuous process: laws, software and the context within which software system operates changes continuously. The works dealing with the compliance problem focus only on one or two subjects: analyze legal documents or extract requirements or identify conflicts or changes. This thesis deals with all the problems at the same time; the idea is to extract requirements from legal text, compare them with the software requirement, resolve the possible conflicts that may arise, continuously leading with the changes on environment, laws and requirements. For this, this work proposes a framework that is composed of a compliance process and continuous monitoring of environmental changes. The framework deals with different types of laws (security, privacy, transparency, health care) that are represented in explicit norms. The compliance process supports the identification, extraction, comparison and conflict resolution to help software compliance, by producing a compliant set of requirements. The compliance process is based on the semantic annotation and goal model. The semantic annotation helps to extract requirements from thelaw, using patterns. The goal model is used to help the comparison between requirement and to represent requirements in a formal and consistent requirement specification. The process is tool supported; some tools were reused (Desiree and NomosT) to further each step. It was necessary to adapt the tools for the context of the compliance process, creating a guideline, patterns, and heuristics. The continuous monitoring is concerned about the changes that affect the software compliance and has

the mechanism to ensure that even with those changes the software will regain compliance. The compliance monitor is basedon agents and Non Functional Requirements. The agents are represented using in i*, the idea is to showthe collaboration between the agents to ensure the continuous compliance. The requirement specification of how each agent should behave was also generated using Business Process Modeling Notation and Desiree language. The Non Functional Requirements catalogue is used to help to define operalizations for the software awareness. The framework validation was made in two parts: first, the compliance process and after all the framework proposed. For the compliance process, the effort and correctness were measured comparing the use of the proposed process andan ad-hoc method. For the entire framework, the example of monitoring the changes in the environment when an automated car is crossing the border between Washington and Canada was used. The study shows that context has a strong influence on the software requirements, and nonconformity problems may incur penalties. The contribution of this work is the Eunomia framework that has a process and goal model perspective with emphasis on monitoring that helps to deal with the compliance challenge. The framework equips the requirements engineering team with a systematic method. Eunomia framework is a tool-supported and systematic process which can be reused to reduce the time effort and to improve the quality of the requirement specification that helps to create a compliant software requirement specification that is compliant over the time.

Keywords

Tool-supported compliance process; software systems design; software system compliance; compliance analysis; requirement compliance.

Resumo

Engiel, Priscilla; Sampaio do Prado Leite, Julio Cesar (Orientador); Mylopoulos, John (Co-orientador). **Eunomia: Um Framework de Conformidade Continua para Sistemas de Software baseado na Engenharia de Requisitos**, Rio de Janeiro, 2018, 141p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Leis e regulamentos afetam o desenvolvimento de software, já que freqüentemente exigem mudanças nos requisitos de software para proteger indivíduos e empresas em relação à segurança, privacidade, governança, sustentabilidade e muito mais. Requisitos legais podem ditar novos requisitos ou restringir os existentes. O problema da conformidade de software é como garantir que o software esteja em conformidade com as normas impostas pela legislação. O problema é particularmente desafiador porque combina etapas difíceis: 1) analisar documentos legais, 2) extrair requisitos desses documentos, 3) identificar requisitos conflitantes com aqueles já implementados em software e 4) garantir que o software permaneça compatível mesmo com as alterações. A conformidade é um processo contínuo: as leis, o software e o contexto no qual o sistema de software opera mudam continuamente. Os trabalhos que lidam com o problema de conformidade concentram-se apenas em um ou dois assuntos: analisar documentos legais ou extrair requisitos ou identificar conflitos ou mudanças. Esta tese trata de todos os problemas ao mesmo tempo; a ideia é extrair requisitos do texto legal, compará-los com o requisito de software, resolver os possíveis conflitos que possam surgir, lidando continuamente as mudanças no ambiente, leis e requisitos. Para tanto, este trabalho propõe um framework que é composto por um processo de compliance e monitoramento contínuo das mudanças ambientais. O processo de conformidade suporta a identificação, extração, comparação e resolução de conflitos para ajudar na conformidade do software, produzindo um conjunto conforme de requisitos. O processo de conformidade é baseado na anotação semântica e no modelo de meta. A anotação semântica ajuda a extrair requisitos do arquivo, usando padrões. O modelo de meta é usado para ajudar na comparação entre requisitos e representar requisitos em uma especificação de requisitos formal e consistente. O processo é suportado por ferramentas; sendo algumas reutilizadas (Desiree e NomosT) para avançar cada etapa. Foi necessário adaptar as ferramentas para o contexto do processo de conformidade, criando diretrizes, padrões e heurísticas. O monitoramento contínuo está preocupado com as mudanças que afetam a conformidade

do software e tem o mecanismo para garantir que, mesmo com essas mudanças, o software recupere a conformidade. O monitoramento da conformidade é baseado em agentes e requisitos não funcionais. Os agentes são representados usando em i *, a idéia é mostrar a colaboração entre os agentes para garantir a conformidade contínua. A especificação de requisitos de como cada agente deve se comportar também foi gerada usando a linguagem Desiree e BPMN. O catálogo de Requisitos Não Funcionais é usado para ajudar a definir as operações para o reconhecimento de software. A validação do framework foi feita em duas partes: primeiro, o processo de compliance e após todo o framework proposto. Para o processo de conformidade, o esforço e a exatidão foram medidos comparando o uso do processo proposto e um método ad-hoc. Para todo o framework, foi usado o exemplo de monitoramento das mudanças no ambiente quando um carro automatizado cruza a fronteira entre Washington e o Canadá. A contribuição deste trabalho é a estrutura da Eunomia, que tem uma perspectiva de processo e modelo de metas, com ênfase no monitoramento que ajuda a lidar com o desafio da conformidade. O framework equipa a equipe de engenharia de requisitos com um método sistemático e suportado por ferramentas que pode ser reutilizado para reduzir o esforço de tempo e melhorar a qualidade da especificação de requisitos.

Palavras-chave

Processo de compliance; desenho de software; conformidade legal; conformidade de sistemas.

Table of contents

List of tables and figures

1
Introduction

During the last years, some efforts have strengthened the compliance issue in the organization scenario: the society longs for trust and ethical organizations, and the government tightens surveillance on compliance with legal aspects involving regulatory policies, regulations and laws applying severe penalties (mainly financial) if licit determinations are not met. The violation of legal and regulatory compliance is affecting large enterprises, especially banks and public institutions, (Millman & Subenfeld, 2014) resulting in a lack of customers´ trust, bad publicity, financial penalties, and sometimes criminal charges (Otto & Anton, 2007).

In Brazil, for example, the organizations with share trades in NYSE must follow the Sarbanes-Oxley (SOX) law. Organizations in the telecommunications sector need to be compliant with ANATEL regularizations and the electric sector with ANEEL ones. Another example is the Internet Civil Mark that determines aset of laws and patterns to the Brazilian internet sector.

Compliance is in growing demand. One example is what happens with Facebook in Russia. In September 2017, the Russian president declared that Russia would block the access to Facebook unless the social network complied with a law that requires websites, which store the personal data of Russian citizens to do so only on Russian servers[1]. In November, the LinkedIn's website had the access blocked to comply with a court ruling that found the social networking firm guilty of violating the same data-storage law[2].

Because of the Brazilian corruption scandal, involving from the oil companies to the country's largest construction companies, the world compliance became more well-known in the country. The scandal has led to stricter rules against corporate irregularities, with the creation of a new anti-corruption law that punishes companies for acts of corruption against public administration. The companies responsible for illegal practices will now pay a fine of up to 20% of their billing.

In the past, compliance was the responsibility of auditors. Organizations are reactive, which only position themselves after pointing out problems and deviations. This is rapidly changing; compliance now is a demand for a software system, the software requirements must be related to the needs of the customers as well as the regulations imposed. Compliance needs to be a proactive situation where the organizations are adopting procedures that ensure compliance with their processes with regulatory requirements.

To be compliant means that an organization satisfies the requirements of the relevant regulations. Compliance often imposes IT controls that focus on information creation and retention, as well as on its protection, integrity and availability. This is an essential issue in many organizations because non-compliance might lead to penalties and reputation-related risks. Given that norms can come from multiple jurisdictions, many of

[1]http://fortune.com/2017/09/26/facebook-russia-political-ads/ Access on July 3rd, 2018.
[2]http://www.bbc.com/news/technology-38014501 Access on July 3rd, 2018.

which are updated periodically, it makes it very difficult for organizations to keep track of relevant legal requirements.

Laws and regulations inform what should be done without providing all the implementation details (on how it should be done). When addressing the issue of compliance with a new or updated law, an organization must interpret the text and find the requirements that affect the organization. To extract the regulatory requirements from the law text, first is necessary to identify where those information are and then understand the impact of that information and the software requirements that need to be implemented. Even though laws are written in a specific language with technical terms and legal jargon –what makesthis work extremely difficult (Kiyavitskaya, Krausová & Zannone, 2008).

For this reason, compliance needs to be implemented jointly by different types of professionals, since it involves a complex scope. Regulations are typically specified in a legal language that is different from the requirement language, for this requirement engineer need to translate the law into software requirements (Zdun, Bener & Olalia-Carin, 2012). The analysis of laws is time-consuming and error-prone, and it must be done by a lawyer specialized in the field. Also, the legal language is hugely different from the computer language being highly difficult to convert legal rights or obligations into organizational and system requirements (Guarda & Zannone, 2008). Requirement´ engineers must work with business experts to ensure that their software and systems are in legal compliance.

To increase the complexity, the law, the requirements and the software context are continuously changing, and this affects legislation governing it. Brand new legislation arises, or old legislations are updated and the requirements to be met by organizations may change consequently. Also, requirements can change very frequently: It can be updated,or unknown requirements can arise and need to continue adequate with the legislation. Last or not least, if the jurisdiction changes, the requirements need to be adapted too, for example, if the softwareis located in Italy, the rules to be followed are the Italian ones, if the country changes, the rules will change too. With technological advances (cloud computing, mobile computing), these context changes are frequent and very fast. An organization must, therefore, ensure that software’s are updated and consistent concerning every applicable law independent of the context that the software is running in and the changes that could happen in legislation and software requirements.

An example of context change that interferes with software requirements is Netflix (a global provider of streaming movies and television series). The content of the website changes when the country changes. In Brazil, we have some legislation that restricts some content based on ages (free, over 10, over 12, over 14, over 16 and over 18). In the USA, there is COPPA (Children's Online Privacy Protection Rule) that imposes specific requirements on operators of websites or online services directed to children under 13 years of age. Therefore, for each country that Netflix works, it is necessary to comply with different norms and to change the site´s content.

In practice, however, the task to discover regulatory requirements on law can be challenging and arduous (Nekvi & Madhavji, 2015). Laws arewritten in legal language and some of them are inherently vast consisting of hundreds of pages of text. Understanding how a software-intensive system may impactorganizational risks

concerning laws and regulations is a challenge since software also tends to be voluminous.

1.1. Motivation

Compliance should be defined internally and constantly allowed to evolve as a business and legislation´s change. When designing a novel system, it is becoming necessary to demonstrate that the system complies with applicable legislation. Similarly, given an existing running system and a brand new law coming, it is essential to evaluate its compliance and adapt the system design accordingly.

There are some works that address subsets of these factors that help to ensure continuous compliance (Maxwell, Anton, 2010; Breaux, Vail & Anton, 2006; Schmidt, Anton & Earp,2012; Massey et al. 2014; Massey et al 2010; Maxwell & Anton, 2010; Antón, Earp & Carter, 2013; Yu, Shinpei & Motoshi 2017; Ghanavati et al.,2014; Flowerday & von Solms 2005; KPMG, 2008; Armellin et al., 2011; Galán et al., 2016; Linh et al., 2015; Namiri & Stojanovic, 2007; Santos et al., 2012; Ingolfo & Souza, 2013; Siegbahn & Engel, 2009), but none addresses all factors. That is the purpose of this thesis: To improve compliance with software requirements and legislation in the scenario where the legislation, the context, and the requirements can change, given a pre-existing set of requirements.

1.2. Research questions and goals

Therefore, the problem that this work aims to address is:

How to ensure continuous compliance between the software and the legislation?

The goal of this work is to: **Improve compliance with software requirements and legislation in the scenariowhere the legislation, the context and the requirements can change, given a pre-existing set of requirements.**

Figure 4shows the compliance challenge. In one side, we have the law, with hundreds of pages written in legal text; on the other side, we have the software and the software requirements written in a technical language. To increase the challenge, we have an environmentthat is in constant change.

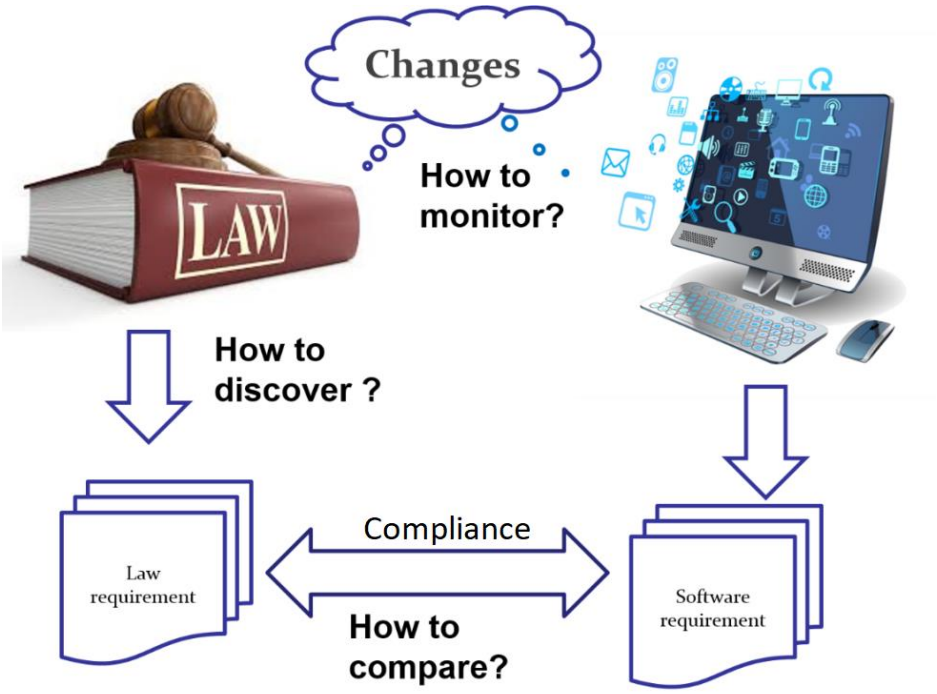

Figure 4- Challenge of compliance

In the light of this work´ goal, we identified different research questions considering the challenges that this work intends to help solve.

The first challenge is related to how to elicit the requirements from the legal text. This is an elicitation problem, whichneeds to deal with text, written in alanguage, whichis not easy to understand for lay people in law subject. Since requirement engineersare not used to this kind of language, our first research question is: **How can requirement engineers extract regulatory requirements from laws in a systematic way?** This question aims to address the problem for all kinds of law subjects (privacy, security, transparency, civil code). The input will be a given law text (or many law texts), by law or business expert, and will return the regulatory requirements that are implicit in law text.

After the regulatory requirements are elicitated we need to compare them with the given software requirements. This is an additional obstacle, because the regulatory requirement will be elicitatedin a different language of the software requirement, so our second research question is: **How can requirement engineers verify software requirements against regulatory requirements in a systematic way?**

When the software requirement is verified against regulatory requirements, conflicts, ambiguities and new requirements can arise. Brand new requirements need tobe added in software specification, although some regulatory requirements can generate conflicts or ambiguities with the software requirement, and because of this is necessary to have methods to identify which software requirements should be modified to make the system compliant. So our third research question is: **How can requirement engineers resolve possible conflicts between regulatory and software requirements?**

The software requirements, the law and the context where the software is running in can change. If one of those situations occurs, the software is not compliant anymore, so it is necessary to have mechanisms that ensure that the software remains compliant. Because of this, our fourth research question is: **How can requirement engineers monitor changes that impact software compliance in a systematic way?** In this case, we will focus on requirement time: how the system needs to work to ensure that requirements are always compliant. We will focus on changes in requirements, law and the software context that affects law (compliance context).

1.3. Contributions

This work aims to ensure that the functionality of the software system being developed will be compliant with relevant regulations considering the changes that may occur and affect the software compliance.

The contribution of this work is to propose and evaluate the Eunomia Framework. Eunomia in the Greek mythology is the daughter of Zeus and Themis, goddess of law and legislation. The name can be translated as "good order," "governance according to good laws." The Eunomia Framework is composed by the Compliance Process and the Compliance Monitor.

The Compliance Process aims to elicit the requirements from the law, compare them with the software requirements and resolve the conflicts that may arise. The regulatory requirements and the software requirements must agree, resulting in a formal, consistent and compliant specification. The work interprets the law as a requirement and verifies if those requirements are already considered in software specification.

It is proposed the use of support tools to help different steps of the process. For this, it was re-used some work already done in the area (Desiree & NomosT), creating mechanisms and steps to link the languages, and adapting the works to the compliance issue. We use a modeling tool for the law, NomosT (Zeni et al., 2016), that the author of this thesis participated on the conception, as well as a requirement´s tool, Desiree Framework (Li et al. 2015), to refine and analyze requirements.

The Compliance Monitor is based on agents that aim to ensure that the changes in the environment, law and software are being observed. When a change happens, the agents act to maintain the software compliance. The contributions are the compliance monitor and the agent specification. The monitoring is performed through a context-aware process that targets law and requirements´ changes based on context-aware catalogue (Cunha, 2014), agents and the concept of product line software requirements to deal with the law variability.

Our proposal is anchored in the belief of the compliance process as a task for requirement´s engineers and has to be addressed at the requirements´ level. The process proposed to define compliance in the software design; we assume that the executable code is compliant with the requirements –the code is doing what it must do. Also, we assume that the requirement is aligned with the organizational business process. The proposal

focuses on Requirement Engineering activities and aims to produce a compliant requirement´s specification.

1.4. Thesis structure

To support the reading of this thesis, avoiding ambiguities and redundancies, we define some terms that must be known by the reader. The term ***regulatory compliance*** means adherence to standards, regulations and other requirements. ***To be compliant*** means that an organization satisfies the requirements of the regulations imposed on it. The ***regulatory requirements*** are related to the requirements that were elicitated from the law text. The ***software requirements*** are the requirements present in software specification.

The text is organized as follows: Section 2 presents the related work of our research; we divided them into four groups (1-elicit requirement from the law, 2- goal modeling and compliance; 3- continuous compliance; 4-compliance monitor process) to facilitate the reading and the discussion since we present different topics. Section 3 presents the research baseline: the tools (NomosT) and frameworks (Desiree, i*, and Context Awareness Catalogue) that we re-use in this work. Section 4 presents the Eunomia Framework: the Compliance Process (the regulatory requirement elicitation from the law, the comparison of them with the given software requirement and the conflict resolution that may be necessary) and the Monitor Process (ensuring the system will remain continuous compliant even with the changes in the environment, law or in software requirements).Both parts together aim to promote the Continuous Compliance. Section 5 demonstrates Eunomia with a preliminary case study for the compliance process using the Italian law and some examples of compliance monitoring, showing how the framework works. Section 6 concludes, highlighting the contributions and indicating future work.

2
State of art of requirements' compliance

In this chapter, we present the state of art ofrequirements' compliance. We divided the related work into fivegroups: 1) requirement elicitation; 2) compliance with legislation and software requirements; 3) goal model and compliance; 4) Continuous compliance; and 5) compliance monitoring. At the end of the chapter, we summarize our contribution related to the other works. This chapter aims to clarify the problem that we want to solve and the existing gaps.

Software compliance is particularly challenging because it combines the difficulties of analyzing legal documents and find the regulatory requirements. To increase the complexity after extracting the regulatory requirements is necessary to compare two set of requirements indifferent languages. Several approaches have been proposed on how to extractsrequirements from the law (Maxwell, Anton, 2010; Breaux, Vail& Anton, 2006; Schmidt, Anton, Earp,2012; Massey et al. 2014). Other works (Ghanavati, Amyot, 2009; Businska et al., 2012;Massey at al 2010; Maxwell, Anton, 2010; Antón, Earp, Carter, 2013) propose how to guarantee compliance between business processes and law (Ghanavati, Amyot, 2009; Businska et al., 2012) or between law and software requirements (Massey at al 2010; Maxwell, Anton, 2010; Antón, Earp, Carter, 2013). Since we are dealing with continuous compliance, we also review the works related to goal model and compliance (Yu, Shinpei, Motoshi 2017) (Ghanavati *et al*.,2014), continuous compliance (Flowerday and von Solms 2005; KPMG, 2008; Armellin*et al*., 2011;Galán *et al*., 2016), and compliance monitoring (Linh *et al*., 2015; Namiri and Stojanovic, 2007; Santos *et al*., 2012; Ingolfo and Souza, 2013; Siegbahn and Engel, 2009).

2.1.
Elicit requirements in law

Breaux and Anton (2005) propose a method called Semantic Parametrization that is used together with goal analysis for the derivation of semantic models of Policy Privacy Documents. To extract the information from privacy policy, they adopt the goal model-driven approach, in which the text is mined,and the resultsarestatements expressed as goals. To mine the text, they use structures like rights, obligations, and constraint and write them in natural language, the natural language is mapped into semantic models to allow the formal analysis.

They also propose a tool that allows a quantitative and qualitative analysis of the model, including the comparison of policy statements. Breaux, Vail, and Anton (2006) combined Semantic Parameterization (Breaux and Anton 2005) with an extended methodology that uses semantic models to find ambiguities and applied the combined methodology in HIPAA (Health Insurance Portability and Accountability Act) Privacy Rule. They extend the framework (Breaux and Anton 2005), including semantic models

derivation from right, obligation and constraint statements using the created patterns. They provide strategies to identify and resolve ambiguities between rights and obligation's exceptions using negation. The work focuses purely in identify ambiguities between the rights and obligations. The conflict analysis proposed is merely in the legal text, they do not compare the law text with system requirements.

Schmidt, Antón and Earp (2012) propose an approach based on goal-based analysis and compare with CPR (Committed, Privileges and Rights) analysis and an ad-hoc one. The CPR analysis (Young 2011; Young and Anton, 2010) is a technique to analyses requirements that help the requirement engineers to extract requirements from policies and data use agreements using committed, privileges and rights. The goal-based analysis has six steps: 1) identify goals; 2) identify agents and responsibilities; 3) eliminate redundancies; 4) obstacle analysis; 5) identify scenarios; and 6) construct goal hierarchy. The CPR analysis consists in three steps: 1) requirement engineer parses the document into individual statements that may be independently analyzed; 2) exam each extract to see if they have committed, privileges and rights and represent them using natural language patterns and heuristics; 3) operationalize the items into requirements using templates. They conclude that a CPR analysis is better than a goal based analysis or an ad hoc one. The work deal with privacy laws, and they do the extraction manually, not having any tool support As our work they use patterns and templates that helps the extraction and writing of the requirements.

Breaux and Schaub (2014) proposed manual methods to extract requirements from privacy policies. Requirement engineer classifies the statements using a code based upon the action type (collection, use or transfer information) and information type (recipients or senders of information's and purpose for which informationis used). They also use modal verbs to find the statements. Crowdsourcing experiments were conducted to evaluate the how untrained workers perform in the task of extracting requirements manually.

These works are concerned with only the first part of the problem that is the requirement's elicitation from the legal text. The methods extract requirement from specific laws (privacy and security) proposing patterns, templates, and structures as right and obligation. Eunomia Framework deals with the compliance between law and requirements So, after the requirement extraction,it is necessary to compare those requirements with software requirements and resolve the possible conflicts that may arise.

2.2. Promote compliance among law and requirements

There are some works, which in addition to finding requirements in the law, also make the comparison with software requirements. Massey *at al*. (2010) verifies the compliance between legal text and software requirements carrying out a mapping between actors, data, and actions that appear in both sets of requirements. As a limitation, the work focuses on verifying ambiguities only in privacy and security requirements. The proposed work is more comprehensive than Massey et al. (2010) since it deals with different kinds of law and checks varied types of conflicts, not only ambiguities.The work also use the structure of actor, verbs, a complement to compare the requirements.

Antón, Earpand Carter (2003) propose an alignment between requirements and policies using inspection. First, they extract the goals from the policies using goal mining and

compare with the requirements. They also propose some heuristics that help to identify ambiguities between requirements. They focus only on privacy and security policies and on discovering goals from the policies. Using the text artifacts, they extract pre-requirement to help the privacy policies analyses. Then they do a cross-examination of the two documents, the pre-requirements and requirement specification. The examination check for terminology conflicts check, ambiguities, incomplete ambiguity (terms left out of the documentation), potential, definite. In our work, we generalized for different types of laws and policies and generated a requirement's specification that is more precise than a set of goals.

Maxwell and Anton (2010) claim that communication between software engineers and legal consultants can generate misunderstandings and lack of compliance. To mitigate this, they use production rules to check the software requirements for legal compliance. Using the query of the production rule model, it is possible to identify non-compliance requirements and specify new requirements to address the gaps. They use as a basisthe search for requirements in law using patterns about rights and obligations as our work does. For each type of law, it is necessary to create new heuristics to elicit the regulatory requirements. They identify non-compliance requirements, but the method does not specify how to deal with them.

Nakamura et al. (2015) propose a technique to support matching the words in a requirement document and regulation for checking compliance. They analyze co-occurrences of words in use in case descriptions and identify the semantic concepts that they have, and also check if they have a compatible concept in regulatory statements. Eunomia proposes to use the help from a law specialist to have a list of the co-related terms in law and the software requirement helping the comparison between the two sets of requirements.

Armellin et al. (2011) propose an argumentation based process as a theoretical solution to the compliance problem between software requirements and law. As we do in our work, they also use the Nomos Framework (Ingolfo et al., 2013). They start generating a Nomos Model for the law.Then, they have verified intentional compliance and compliance audibility. Intentional compliance is verified if every actor fulfills its goals, the compliance audibility is the capability of monitor the solution to confirm compliance (if a task is not executed or executed incorrectly, the process is not compliant. A. The last step is to change the requirements according to the decisions taken for increasing compliance (adding data log resources to goals, adding missing compliance goals). Differently, fromEunomiathey use argumentationto analyze compliance between law and requirement, and Eunomiasuggests a comparison. Eunomia provides a list of heuristics to help the comparison.

2.3. Goal models

Yu, Shinpei and Motoshi (2017) propose a technique to detect the potential's violations of regulatory and system goals. To the find violation, they use matching goal's descriptions and some patterns. Then, to resolve it they propose to add new goals, and never delete an existing one (to maintain the traceability with the father goal). They do not explain how to extract the goals from the regulatory statements and focus on the part of detecting and solve the violations. Our work also models the requirements using the goal model approach. However, this thesis uses a different approach to find and resolve

the conflicts. We aim to modify the requirements to solve the conflicts, but in the last case, is proposed to delete the requirements that are not compliant, although the tool helps to keep all the traceability of the changes.

Ghanavati et al. (2014) propose an extension to the goal-oriented modeling LEGAL –URN (User Requirements Notation) framework that helps to promote the compliance with multiple regulations. They convert the legal URN model, the legal requirements and business requirements in the same notation. They use the goal model to capture main objectives and the structure of legal requirements. The framework aims to model regulations in the same notation as organizational goals, objectives and business processes using Hohfeldian statements (modal verb, clause, preconditions, and subject). Different from our proposal they focus on the business objectives and business model not in software requirements. NomosT tool also based on Hohfeldian statements, and we have same heuristics to do the mapping between law and requirement.

2.4. Continuous Compliance

In the past, compliance was the responsibility of auditors. The majority of papers propose continuous auditing (Flowerday and von Solms, 2005; KPMG, 2008). The continuous compliance is brand new, and since just ensuring compliance is already a challenge, few jobs are concerned with changes that affect compliance – continuous compliance.

Galán et al. (2016) propose an adaptive compliance process that comprises 4steps: (a) Discover new compliance source (e.g., new regulation that applies to the system) or other change to the environment (e.g., new jurisdictions) that calls for new compliance; (b) Interpret new requirements; (c) Implement; (d) Evaluate degree of compliance. The authors also conducted a literature review of software compliance to see how the literature can support the proposed process. This work also proposes a compliance process butintroduces tools that help to vary degrees these steps. Galán et al. (2016) did a compliance literature review considering run-time system; we considercompliance in RE activities.

2.5. Compliance monitor

The majority of works thatdeal with Compliance Monitor focus on establishing compliance in business process model scope (Linh et al., 2015; Namiri and Stojanovic, 2007; Santos et al., 2012). The works that focus on requirements (Ingolfo and Souza, 2013; Siegbahn and Engel, 2009) are concerned in only one aspect of change the environment.

Ingolfo and Souza (2013) propose that existing techniques for the design of adaptive systems can be used to help the need to adapt to legal requirements. In particular, they use the Zanshin framework (Silva et al., 2011) and focus on Legal Awareness Requirements (LAwReqs), the class of legal requirements that lead to feedback loop functionalities. LAwReqs are related to the states that legal requirements can assume at runtime. The proposal uses always the same set of LawReqs. Some of these legal requirements are awareness requirements, so in the runtime that can have different behaviors. Eunomia is concerned with the changes in the environment that influences the

legislation that governs the software, changing the set of requirements that the software needs to run.

Regarding context and law, Stieghahn and Engel (2009) say that laws can cause conditional constraints that are related to context information but cannot be represented by one of the contexts of the context model. Such a conditionalconstraint could be a signed customer agreement. They propose a schema of a legislation government access control model that shows how the legislation can affect the control access governing the legal data. Differently, from Eunomia, they do not propose a way to help the system to stay compliant in distinct context situations.

In the scope of monitoring the business process (Linh et al., 2015; Namiri and Stojanovic, 2007; Santos et al., 2012) there are different approaches: some focus in design time, other in compliance checking architectures or post-mortem conformance checking. As our focus is compliance with requirements, we do not focus on those works. Even so, in business process compliance monitor most approaches are not supported by software tools.

2.6. Comparison among related work

We summarized the comparison of our work (Eunomia), in the last column, with the most similar and present in (table 1).

Table 1 Related work comparison

	Massey et al. (2010)	Ánton, Earp, and Carter (2003)	Maxwell and Anton (2010)	Galán et al. (2016)	Ingolfo and Souza (2013)	Eunomia
Extract requirements	No	Only goals	Yes	Yes	Yes	Yes
Different law domains	Only privacy and security	Only privacy and security	Each law – new heuristics	Yes	Yes	Yes
Discover conflicts	Only ambiguities	Only ambiguities	No	No	No	Yes
Tool-supported	No	Yes	Yes	No	Yes	Yes
Requirement atime	Yes	Yes	Yes	Run-time	Run-time	Yes
Changes in requirements and law	No	No	No	Yes	Yes	Yes
Changes in a lawcontext	No	No	No	yes	No	Yes

Table 2 shows that the related works focus on a particular aspect ofthe problem: only on elicitation, or comparison, or part of conflict resolution, or monitoring. The thesis intends to have a more general compliance process that can be applied to different law domains and includes the extraction, comparison and conflict resolution.

The thesis focuses on Requirement Engineering activities and aims to produce a compliant requirement´s specification. Therefore, the process is tool-supported. For this, we reuse some work already done in the area (Desiree and NomosT) creating mechanisms and steps to link the languages and adapting the works to the compliance issue. Also, in the literature review, was observed that the continuous compliance is still a very new topic.

2.7. Chapter summary

This chapter presented the related work that represents the state of art of Contin uous Compliance Process. We also compared the related work with our proposal to sho w the similarities and the differences.

The chapter presents the lack of support for continuous compliance monitoring. Most of the related work focuses on one aspect, the discovery of regulatory requirements or the comparison of regulatory requirements with software requirements. Some ideas of the presented proposals have been improvedin this thesis. The next chapter we will present the Eunomia Framework, that intends to have a tool support process to help to promote the continuous compliance.

3
Research baseline

This section presents the NomosT framework for semantic annotation of legal documents, the Desiree framework – a tool for systematically transforming stakeholder´ requirements into a formal and consistent requirements specification, the i* language, that helps to model actors intentions and their dependencies,besidesthe Context Awareness Catalogue – a catalogue that aims to store in an organized way knowledge about consciousness that maybe would be used by developers in cases where the requirement for consciousness is necessary.

3.1.
NomosT framework

NomosT (Zeni et al., 2016) supports the semi-automatic generation of models that represent law from legal documents. NomosT is based on the Nòmos modeling language (Ingolfo et al., 2013) andon theGaiusT framework (Zeni et al., 2013), which uses structural and semantic patterns to extract and annotate fragments of texts in legal documents.

We decided to use NomosT because we had to extract requirements from legal documents. There is a lock of tools that extract concepts of law with semantic relationships among them (Zeni et al., 2016). The other approaches (Moens,Uyttendaele &Uyttendaele, 1999; Biagioli et al., 1999; Agnoloni &Tiscornia, 2010; Dozier et al, 2010; Wyner &Peters, 2011; Robaldo et al., 2014) share techniques used for semantic annotation, syntactic patterns and model elements, although none of the other tools proposed dealsexplicitly with the problem of tool-supported generation of full-fledged models of law.

The NomosT framework uses a combination of patterns and regular expressions to identify concepts and the relationships of the law text, allowing us to create new patterns, and new syntactic and semantic indicators to adapt the tool for each necessity.

The core component of NomosT Framework uses and manipulates Nòmos concepts, grounded on five concepts namely:

- Role: the subject of a norm;
- Norm:an atomic fragment of law. The norm can be a
    - Duty (what must be done – an obligation) or a
    - Right (an entitlement) and
- Situation: a state of affairs where some propositions are true, others false and some neither true nor false

Nomos uses the following relations between Norms and Situations: Activate, Block, Break and Satisfy. Activate and Block relates situations to norms and determines the applicability of a norm, Break and Satisfy determine the satisfiability of a norm or situation. A Norm can be the antecedent or a consequent of a Situation.

The relationships between Role and Norms are: A Role can be a beneficiary of the norm who profits from the norm or a holder, a role responsible for complying with the norm. Figure 2 presents the meta-model of Nomos2.

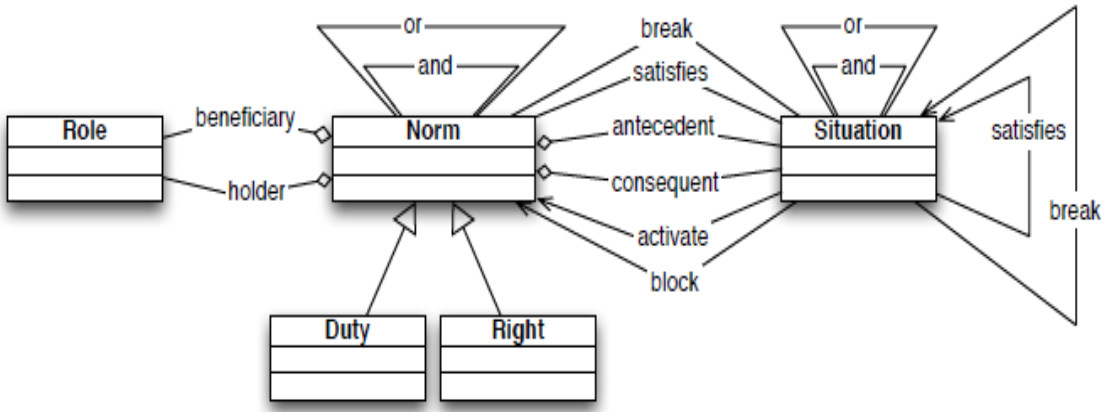

Figure 2 Nòmos 2 meta-model (Ingolfo et al. 2013)

In the example "The data controller must process genetic, biometric or other data," the "data controller" is the Role, which in this case is a holder (need to fulfill the on of genetic and biometric process or other data).

To extract the Nomos concepts from law using NomosT tool,a set of auxiliary patterns (Actor, Resource, Exception, Antecedent, Consequent, Positive Verb, Negative Verb, Holder and Beneficiary)was created. A pattern can be nested within other patterns. For example, the Role concept has syntactic indicators that identify instances of an Actor, Beneficiary and Holder while the pattern for the concept of Situation is a combination of an Actor or Role, an Action and a Resource.

The patterns use the concept of modal verbs to represent right and obligations. Positive Verb identifies instances of a modal verb that express permission (may, can, could), while Negative Verb identifies instances of obligation and prohibition (must, should, cannot, must not).

Table1shows some of these patterns. Concepts, patterns and relationships are described in the annotation schema and are the basis for semantic annotation and the generation of Nòmos models. The output representing the generated model is coded in XML.

Table1NomosT patterns

Concept	**Pattern**	**Example**
Right	(Right):: = ((Actor) | (Role)) + (PositiveVerb) +(Action) + (Resource))	The guarantee may specify additional processing operations that are liable to affect the data subjects' rights
Obligation	(Obligation):: = ((Actor) | (Role)) + (NegativeVerb) + (Action) + (Resource))	A data controller shall notify the processing of personal data

Actor	(Actor) :: = (Person)|(Subject))	The Guarantee, Data controller,
Action	(Action) :: = (Verb)	Specify, notify
Resources	(Resources) :: = (Complements)	Additional processing operations that are liable to affect the data subjects' rights

NomosT supports requirements´ discovery in law. However, the NomosT primary goal is to generate semantic models of law from regulatory documents. Figure shows an example of a semantic model that was generated using NomosT from an extract of Italian Privacy Law.

Figure 3NomosT semantic model example

After extracting the requirements from the law (the regulatory requirements), we need to compare them with software requirements to detect conflicts, and for this, we will use the Desiree Framework that helps to suggestthe lack of compliance. The Desiree Framework is presented in the next Section.

3.2. Desiree framework

Desiree (Li et al. 2015) helps to transform stakeholder´ requirements that are inherently informal, vague and conflicting into a set of well-defined requirement´ specifications. The core of the framework is a set of requirements´ refinement operators who interactively transform stakeholder requirements by strengthening or weakening them, thereby diminishing incompleteness, removing ambiguities and vagueness, contributing to the reduction of conflicts. In this work, we chose to represent the regulatory requirements and software requirements using the same language to compare them and to check for conflicts, so we decided to use the Desiree Framework.

We chose the Desiree framework because we need to compare two sets of requirements to establish compliance. Desiree framework uses a unified language for representing both Functional and Non-Functional Requirements, to capture the interrelation based on requirement details and to have mechanisms to support the requirements conflict´ resolution. Another similar framework that deals with ambiguity and traceability is KAOS (SEMMAK, GNAHO& LALEAU, 2008), although they only apply to goals that can be formalized.

The framework includes an ontology for modeling and classifying requirements, and also a description-based language for representing requirements. We will reuse the example in (Li et al. 2015) to explain the ontology. In particular, the ontology includes the concepts of:

(a) **Goal (G) -** Goals represent stakeholder intentions, which are manifestations of intent, which may or may not be realized. For example, the goal "Manage nurse school" specifies the intention to have a system, which will manage a nursery school and will be represented as G: "Manage nurse school."

(b) **Functional Goal (FG) -**Functional Goal represents the desired state-of-affairs and is operationalized by one or more **Functions (F).** A Function (F) specifies the desired capability and necessary information for completing its manifestation. For example, the goal "student records to be managed" specifies the desired state "managed" that isoperationalized by functions for adding/removing/updating students' records. In the Desiree language this will be represented as FG1 := Student record: Managed; F := Add <object: student data>; F: Remove <object: student data>; F: Update <object: student data>;

(c) A **Function Constraint (FC)** constrains the situation under which a function can be carried out. For example, only managers can activate a debit card, so the function is activated debit cards, and the constraint is that only managers can do it; In the Desiree language this requirement will be represented as FC: Activate <object: debit_card><actor: ONLY managers>;

(d) **Quality Goal (QG) and Quality Constraint (QC)** are requirements that require quality to have value. The Quality goal whose quality regions are vague and quality constraints whose quality regions are specified. Ex: The file search function shall be fast. Fast is the quality constraint for the function of file search; That will be represented as QG: Processing time (File search): Fast and the quality constraint will be QC: Processing time (File search) < 15 Sec;

(e) A **Content Goal (CTG)** and **State Constraint (SC)** often specifies a set of properties of an entity in the real world, including both attributes and qualities, and these properties need to be represented by a system-to-be. To satisfy a CTG, the system-to-be needs to be in a certain state which represents the desired world´ state. That is, concerned properties of real-world entities should be captured as data in the system. We use a state constraint (SC) to specify such a desired system state. For example, to satisfy the CTG "A student shall have Id, name and GPA," the student record database table of the system must include three columns: Id, name and GPA. This example can be captured as a content goal CTG1: = Student :< <has id: ID>< has a name: Name> <has GPA: GPA>and a state constraint, SC2: = Student record: << ID: String> <Name: String> <GPA: Float>.

Desiree Ontology also has relationships. In this work, we use the relationship **ReferTo**, which aims to capture the interrelations among Fs, FCs, QGs, QCs, CTG sand SCs. In general, an F could refer to some CTGs or SCs. For example, we have the requirement of Search for a Product Information, and this search needs to be processed within a maximum time of 30 seconds. In this case we will have a Function F1: Search<object:Product-info> and a Quality Goal QG: Processing_time :: [0,30 (Sec).] that ReferTo this Function.

Desiree also provides a systematic method for applying the refinement operators to transform elicited requirements into a specification. In this work, we will use only the **Resolve (R)** operation, intended to resolve two or more conflicting requirements into one or more non-conflicting ones.

Finally, a graphical modeling tool supports the framework. The graphics editor supports requirements, concepts and operators of the Desiree framework, also allowing engineers to visualize the specification. Figure 2 shows the graphical model in Desiree where functions are represented as diamonds, goals are represented as ellipses and constraints are represented as a square. The red C represents the conflicts and the Rftorefers to the relationship.

The tool also allows the transformation of requirement´ models to OWL2 ontologies (Motik et al. 2009). As such, allowing requirements engineers to perform reasoning tasks such as interrelation query, inconsistency check, and what-if analysis over the resultant requirements´ ontologies.

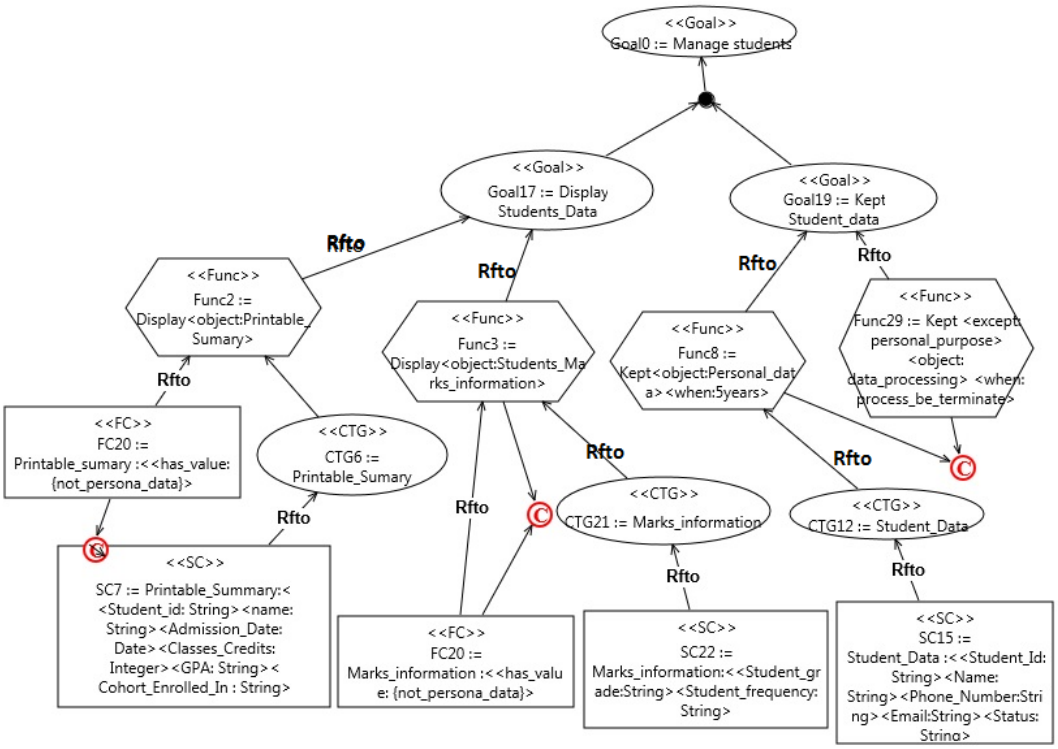

Figure 4 Desiree tree example

3.3. Goal model

Goal models like the NFR Framework or i* (Yu, 1995) provide a way to represent the variability of possible ways to contribute to a goal. The variability is essential to adaptive software that needs to have alternatives to adjust to different situations. Therefore, in this work, the software needs to adapt to the context changes, so we decided to use goal models to represent how the compliance monitoring will work and also the agent's intentionality.

The i* framework (Yu, 1995) models the perspective of an organization´s context based on the actor´s dependency. It is an approach created for modeling and reasoning about organizational environments composed of actors with different or shared goals that depend on each other to achieve their tasks. The i* 2.0 core language (Dalpiaz, Xavier &Horkoff, 2016), involves the core construct and the original symbology of i* although it inserts more simplicity, usability, expressiveness to those constructs as agents and role,besidesthe association´s links, the actors´ boundaries, the tasks, resources, dependencies and some contribution types.

The i* consists of two primary models:

- **Strategic Dependency model (SD):** The model aims tocapture the actor´s intentional structure. The diagram focus on what matters to the actor, describing the actor dependency, using the network dependency. The model helps to observe the opportunities, vulnerabilities and relationships among the different actors. This model shows the actors and the dependency upon the work of a determinedactor. Figure 3 shows an example of anSD model, where the Actor A depends on Actor B to achieve the goalthat Actor A want.

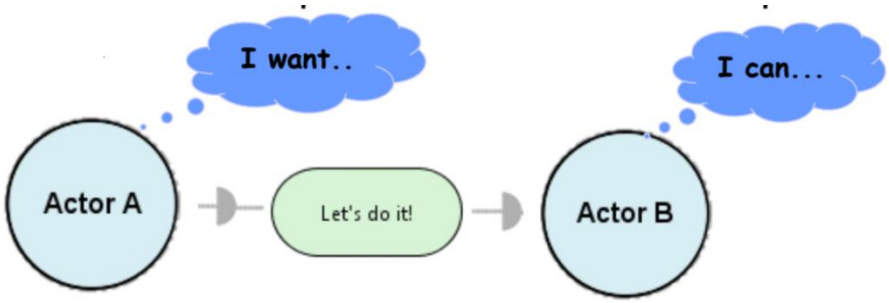

Figure 5 Example SD Model [Yu95]

- **Strategic Rationale Model (SR):** The model describesindetails the actor´s intentions and what he needs to do to achieve his goals, also the rationale behind. The diagram focuses on the intentional relationship using goals (condition or desired state of the world that wants to be achieved), task (particular way to do something), resources (physical or informational unit available) and softgoals (qualities of the world that want to be achieved) to represent what the actor intends. Thereare two relationships: the task decompositions and the means-end link. The diagram uses the relationship means-end to show the means to accomplish the end explaining why and how the alternatives will be achieved. The task decomposition shows the steps to achieve one task.

Figure 6 shows an example of SRdiagram: in this case, we have 3 actors: the the Customer, the Middleman and the Supplier. The Customer aims to purchase by naming his own price, andfor it he needs to name a price he wants in order to find a low price service provider. To find the lowest price, he needs the Middleman, who will search it with the Supplier.

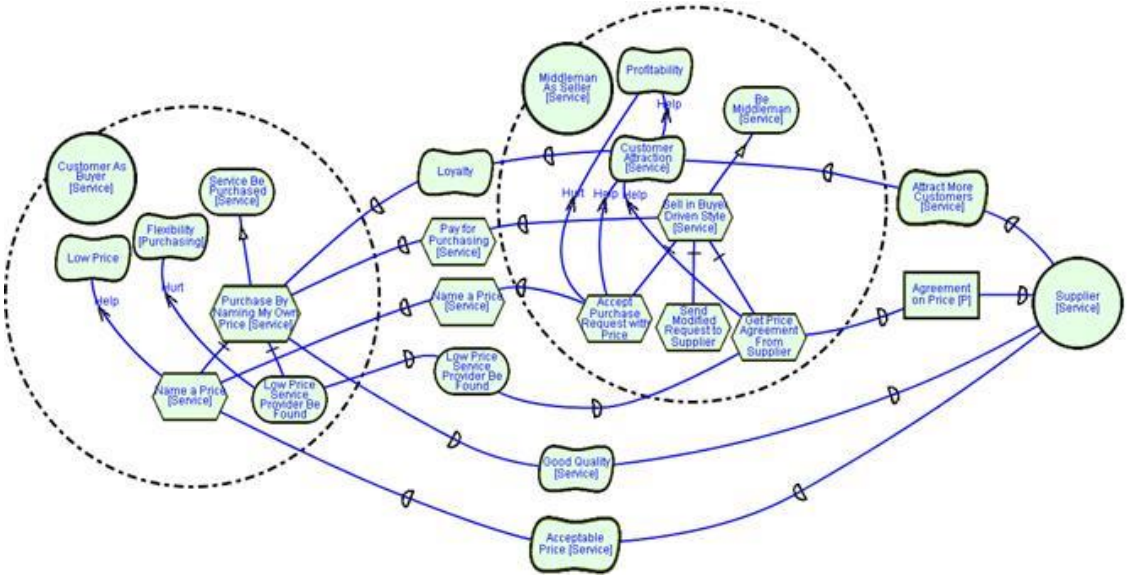

Figure 6 Example SR model from [Yu01]

We will use the SR Model to represent the interaction between the actors in continuous monitoring since we want to show the reasoning and the variabilityavailable during theagent execution.

There areother structures and models that wereproposed based on i*, for example, theStrategic Actor (Leite et al., 2007), SD Situation (Oliveira 07) (Oliveira 08a) and SRConstructs (Oliveira 09b) to add another points of view.

- **Strategic Actor (SA Model**): It is used mainly to model the actor in i*. This model helps to understand who is the actor and his relationships from a structural viewpoint. The SA model uses the same concepts present in SR model: agent, position and the role, also and showing the relationship among them. Figure shows an example of the SA model to represent the actor in the scope of a conference paper review. We have five roles: author, reviewer, coordinator, conference organizer and conflict's solver. The Conference Team Member (agent) is a Researcher that can play the role of Coordinator Author, and Review. He also may assume the position of a Chair and in this specific case, cover the role of Conference Organizer. Although if he covers the position of Committee Member, he can cover the role of Conflict's solver or Reviewers.

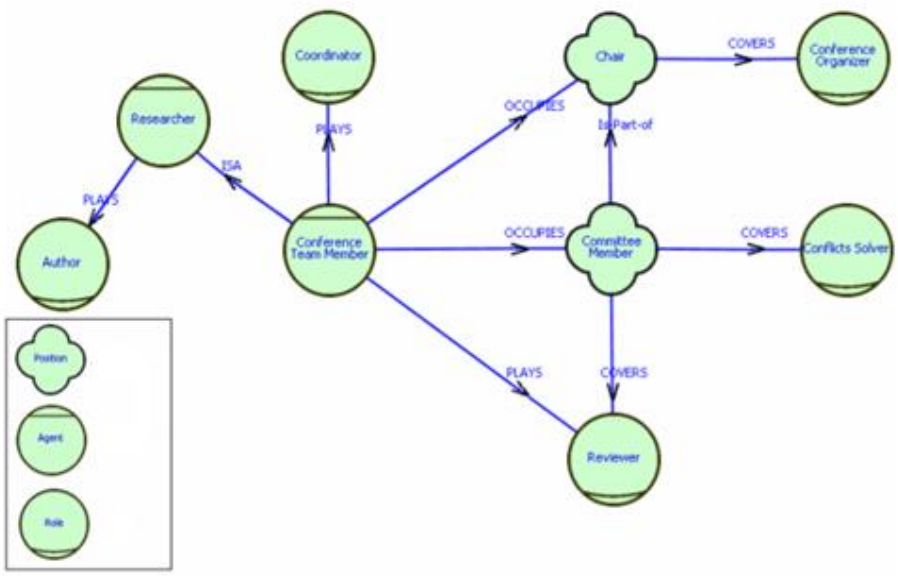

Figure 7Example SA Model [Cunha 2007]

- **SDSituation Model:**It shows the time sequence. The situationsaim to show the order that the tasks will happen to achieve the goal.

  Using the same example of the conference paper review Figure shows the SDsituation. The process starts with the reviewers and the Committee indication. Only when those two tasks are finished and a T1 is over, the article is submitted for a proposal acceptance. After the proposal acceptance, the article´ review starts and if necessary the conflict's solution happens. After these two tasks are finished, the author needs to generate the camera ready.

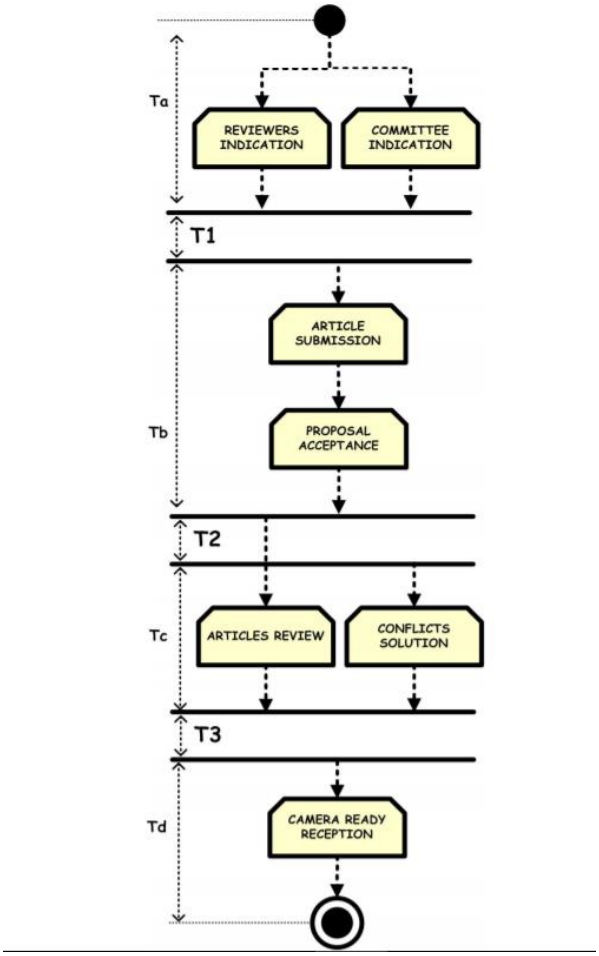

Figure 8 Example of SDsituation model [Cunha 2007]

- **SRconstructs:** SRconstruct is part of the actor rationale that models the necessary elements to achieve the target´s goal from the own SRConstruct. The SRConstructis composedof a goal (“end”), at least one task (as the means to achieve the end) and all the components (or subcomponents) needed for the tasks. The central point of the SRcontruct is that in every unit of a SRconstruct there are the components of the SRcontruct that are connected to help the means to reach the end of the structure. SRConstruct is a canonical structure. A SRConstruct establishes a strategy that can be reused in other cases. We can say that a SRConstruct is a design pattern for i * models. Figure 10 shows a SRConstruct used for the awareness of an agent.

a. Model i* representation in iStarML

iStarML(Cares et al., 2007) is a textual specification that has a compatible format with XML to represent the i* diagrams. The specification was created to:

(i) Have a file format to interchange diagrams between different types of i* software asgoals analyses, drawing, editing and the metric calculation

(ii) Find away to represent the differences and similarities among the existing i* variations;

(iii) Have a standard representation to i* patterns

(iv)Use the XML format for Internet communication and use of general XML tools.

Cunha (2014) uses the iStarML specification to add new elements in i*, allowing the software awareness´ aspects that will be reused in this work.

We will use those three diagrams (SA models, SD Situations Models and reconstruct) to explain how the agents are organized. The situation on how the sequence of the monitor tasks will happen. Finally, the SRConstruct to reuse the design pattern already created by [Cunha 2014] to represent the awareness of the agent since we are modeling the monitor part and the agents need to be aware of the changes in the environment.

3.4. ContextAwareness Catalogue

Deyet al. (2002) define context as "any information that can be used to characterize the situation of an entity. An entity is a user, a place, a physical or computational object that is considered relevant for the interaction between a user and an application, including the user and application themselves."

A system is context-aware if it uses context information to adapt their behavior. Cunha (2014) says that Software Awareness is an important requirement for software that needs to be autoadaptive, since, to adapt, the software needs to perceive and understand the environment and its operation on the environment. To promote the continuous compliance the software will need to adapt, so we use the idea of Context Awareness Catalogue(Cunha 2014), that organizes the knowledge about awareness. The advantages on using Catalogues are to share the knowledge about the non-functional requirement and enable the reuse.

Compliance, to be fully implemented, needs self-adapting systems that keep aware of the changes in requirements, in the law or in the context it operates and reacts to the changes to stay compliant. Cunha (2014) defines context awareness as awareness of the external context that the software is embedded. The software awareness is a non-functional requirement and can be represented as softgoal.

To achieve the awareness softgoals, Cunha (2014) creates some operationalizations. Operationalizations are development techniques that help to satisfy the softgoals.

The awareness´ softgoalsare decomposed into lower-level types or subtypes: Physical Environment, Computing Environment and Location. The operationalization of the higher level type is achieved by the subtype operationalities. In doing so, the task of finding possible operations (solutions) becomes easier. Figure 9 shows this information in the Catalogue (Cunha, 2014).

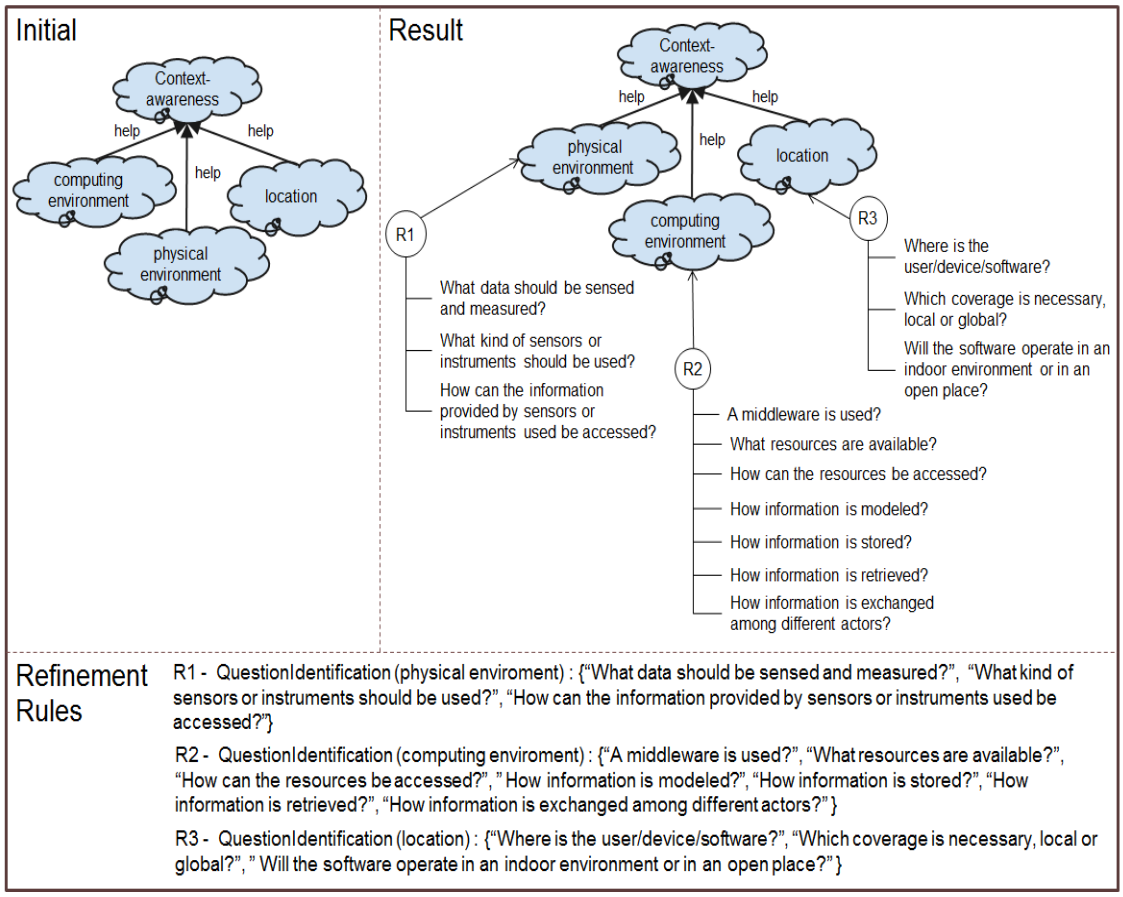

Figure 9 - Context Awareness GQ (Cunha, 2013)

Cunha (2014) uses the Goal-Question-Operationalization (GQO) pattern (Serrano & Leite, 2011) to find the operationalizations for each topic context. We will use this pattern to monitor the environment and discover the changes that will affect compliance.

The questions are essential to guide the search for operationalizations. The questions are presented as question patterns for non-functional requirements, while the operationalizationsarepresented as operationalization patterns of non-functional requirements.

For each subtype of context, Cunha defines questions and operationalizations (Figure 9).

- Physical environment – As a software system becomes pervasive, to be aware of the computing environment is essential. The techniques to operationalize the context awareness share the same core: context monitoring. The following questions are relevant to the environment context: "What data should be sensed and measured?", "What kind of sensors or instruments should be used?,","How can the information provided by sensors or instruments used be accessed?."To answer the questions, two operationalizations are essential: (i) to monitor sensor´s use and (ii) to make use of the communications protocol.

- Computing environment – In ubiquitous computing, as well as distributed computing, an effort is put to build middleware. The middleware will support the exchange of context information. The relevant questions are: "Which middleware

is being used?"; "What resources are available?"; "How can the resources be accessed?"; "How information is modeled?"; "How information is stored?"; "How the information is retrieved?"; "How the information is exchanged among different actors?"To answer to those questions, some techniques are used to describe the context information openly as XML, RDF and OWL. Some infrastructure or middleware as well can help to answer those questions as JASON, JADE, SACI or also patterns to change information as UML, i* or KAOS.

- Location: The question about location is obvious, and is valid for every domain: "Where is the software?"To answer this question, some technologies have been developed, like Position Systems. Those technologies can be classified as global or local. Other questions are "Which coverage is necessary, local or global?"; "Will the software operate in an indoor environment or in an open space?"

Some operationalizations for context awareness are present in the literature. Ferreira, Maiden & Leite, 2015 propose a framework that aims to anticipate some possible impacts of organization changes in the socio-technical system requirements. Another example is a corporate control architecture (UCCA) for Unmanned Aerial Vehicles (UAV), flying robots or drones (Tian et al. 2012).

a. Awareness in i*

To represent the awareness´ requirement in i*, it is necessary to represent how the awareness requirement can be implemented. For this, SRContruct proposes to:

- Represent awareness through context, with their respective situations;
- Allow the operationalization (refinement) of consciousness requirement
- Guide the implementation of the consciousness requirement.

Contexts are directly relatedto the problem. Depending on the context, situations underlying the software operation will occur. Thenawareness requirement is essential, helping software self-adaptation and autonomy at some level. Thekey point is to identify which are these situations that the software should perceive during its operation. Once you identify which situations are relevant in a determinedproblem domain, the following implementation-related questions need to be answered:

I- "What data should be acquired?";

II- "How do I get the data I want?";

III- "How can the acquired data be interpreted?".

The instruments used for data acquisition varies according to the software awareness subtype in question, as well as the mechanisms used to interpret this data varies according to the type of data and the amount of data available. Also, the data that must be acquired varies according to the problem domain. The consciousness requirement operationalization is divided into two integrated steps:

• Data acquisition: a function that in a given context description returns the underlying situation represented by this description at a given time point

• Interpretation of acquired data: function to map context instances acquired in the process of data acquisition with a set of known situations.

An approach to make the interpretation is to define a set of real-world situations that can be captured through data acquisition, and for which software could decide how to (re) act.

For this, Cunha (2014) proposes to add to the i* models abstractions that help the software to perceive the environments and its changes, and relationships to relate these new abstractions to the other elements of the models that represent the software behavior. Then itis necessary to representi* models:

- The context situations: state of the world at a given time or during a time interval in a given environment
- The connection of the context situation with the entities in which they are anchored
- The connection of the context situation with the alternatives to satisfy the goals

As such, i*was extended to include new elements:

- The intentional context element –as a new type of requirement (context awareness).
- The link between the intentional context element and the alternatives means to achieve the goal related to the context. This link (!Situation) is similar to the –NFR Framework argumentation link, used to record the rationale in choosing an alternative.

The ContextAwarenesswas defined as the iStarML tags:

Context awareness: A context is a set of the situation at a given time or over a periodin a given environment. A situation is a state of the world. Context awareness is a special task (mean) to satisfy the awareness NFR (end)

<ielement type=”context-awareness ” />

The approach is divided into two steps: data acquisition and data interpretation. Because of thisthe context-awareness element was decomposed into this two subtasks, data acquisition and data interpretation. The set of situations for the softgoal of “consciousness” together with the context awareness element and the subtasks form the canonical structure known as SRConstruct idealized by Oliveira (Oliveira, 2008).

Figure 10 represents the SRConstruct for the softgoal “consciousness.” The situations are represented by the argumentations links in red, where an exclamation mark (!) precedes the labels. In this way we can bring awareness to an i* model, thus

dealing with the issue using a special construct in the extended i* language (Yu, 1995).

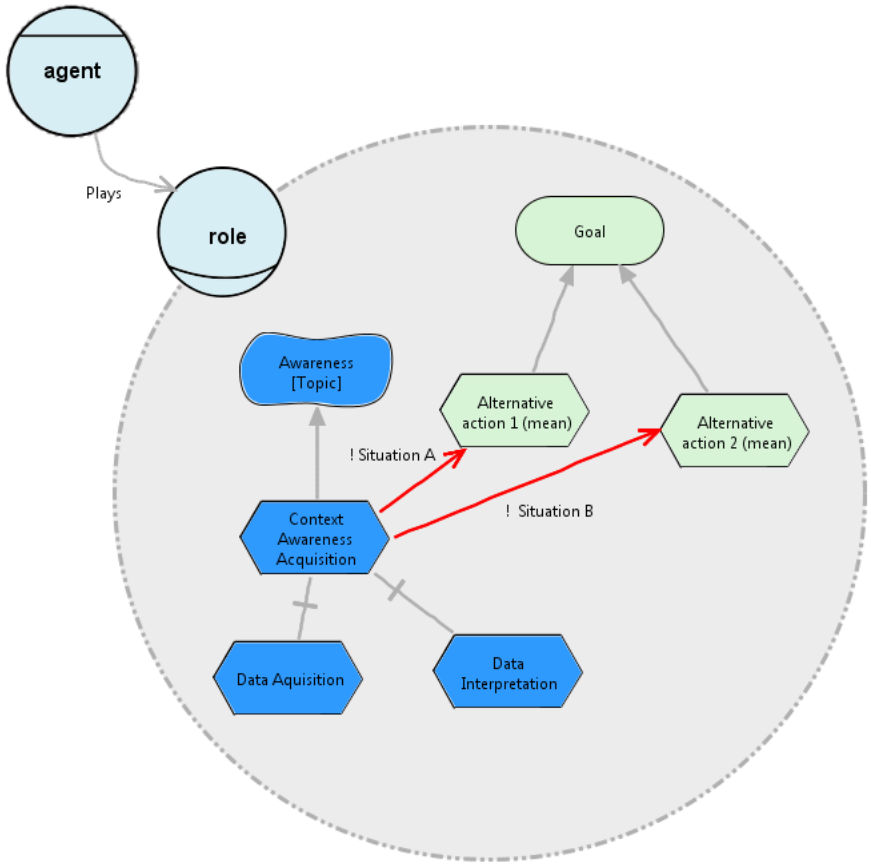

Figure10.SRConstruct for the softgoal “consciousness” (Cunha, 2014)

A key point for the requirement engineering of consciousness is to define the set of situations that belong to a given context. For some classes of problems, the identification of situations and the impact on the context for meeting the goals can be direct and explicit. There are problems where the goal is to monitor some signals to the environment and (re) act by the changes that are the causes of the monitor context. In these classes of problems, the corresponding behavior in response to the underlying context situation can be defined in a deterministic way: based on values of monitoring variables it is possible to decide the best action to be taken in these situations.

3.5. Chapter summary

In this chapter we present the background used inthis research. In the literature, we found some tools and frameworks that help our process supporting thecontinuous compliance.Then we incorporate them for complementing our framework. First, we present the NomosT, a tool based on the Nomos conceptual language that will help us to mine the requirements of the laws. We also need a way to compare the regulatory requirements and resolve conflicts that arise, and for this, we proposed the use ofthe Desiree Framework. In the context monitoring activity, we use the intentional modeling and take advantage of the existing Awareness Catalogue to represent the context monitoring. Our proposal base is presented in the next chapter.

4
Eunomia (Εὐνομία) framework

In this chapter, we present the Eunomia Framework that is composed of the Compliance Process (4.1) and the Compliance Monitoring (4.3). The first aims to promote compliance with legislation and software requirements written in different languages. The second aims to encourage the continuous compliance, monitoring the changes and acting to reconfigure the system to promoteits compliance.

Regulatory requirements are not always known and are not incorporated into the software systems functional and non-functional requirements because regulations are typically specified in a highly abstract legal language that requires experts to read, understand and reinterpret them into detailed requirements. Also, regulations are voluminous, so the task to generate regulatory requirements is arduous. Requirement engineers must work with experts (legal consultants) to promote that the software meets these requirements. Also, the legislation, the requirements and the context of the software can change. Therefore, we need a method that supports the continuous compliance.

4.1.
Conceptual model

To explain the Eunomia framework, we create a conceptual model (Figure 1). The system is composed of the software, and the software is composed of Software Requirement Specification Compliant (SRS Compliant). To be compliant means that software satisfies the requirements of relevant laws.Lawsare described in the legal text. The NomosT tool (Zeni et al., 2016) models the law and helps us to discover the regulatory requirements in the legaltext. The software requirements are compared with regulatory requirements and can generate some requirement conflicts that need to be solved in compliance requirements (requirements that conform with the law). The Desiree tool (Li et al. 2015) helps the modeling of the regulatory requirements and the software requirements in the same representation, helping the comparison of both sets of requirements. In this comparison, some conflicts may arise and are necessary to resolve them to generate the Software Requirement Specification (SRS) Compliant. The SRS Compliant is composed of Software Requirement (from the Software Specification), Regulatory Requirement (from the law) and Compliance Requirements (software requirements that conflicts with regulatory compliance and need to be updated). These compose the Compliance Process (Engiel, Leite & Myloupolos, 2017) represented in blue.

Laws, requirements and the environment can change and affects the regulatory and software requirements. GPS, Communications Protocols and Sensors help to monitor the Compliance Context, generating the environment changes that affects the law. The Compliance Context is composed of Time, User and Jurisdiction. The Jurisdiction depends on the location. The combination of the variation in compliance context generates the situations that the software needs to adapt. For each situation, it is necessary to have a SRS Compliant. The SRS Compliant will have the requirements necessary for

the software to be compliant. Those different SRS will be organized as a product line, where the common requirements for all situations will be in the family requirement. The requirements that are specific for each situation will compose the specific SRS. The software changes happen because of the client's needs and the law changes from the Legal Office needs; those changes will affect the SRS, so the compliance process needs to run again to ensure that the software remains compliant. The Compliance Monitor is represented in green. The Eunomia framework, the compliance monitor and the compliance process are explained in the next sections.

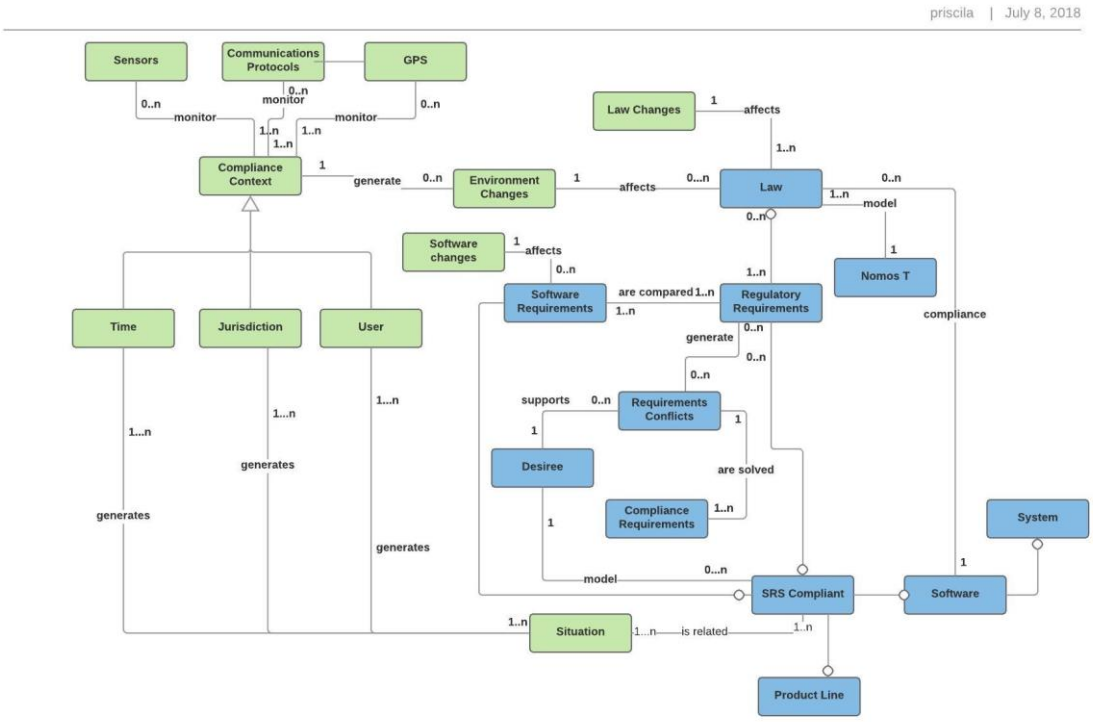

Figure 11 Eunomia conceptual model

4.2. Compliance process

The proposed process (figure 12) elicits requirements from thelaw, compares them to existing software requirements and resolves possible conflicts, using a set of heuristics, resulting in a compliant specification. The overall process is enacted by two different groups of actors: legal consultants and requirements engineers. A legal consultant is a person who knows about the domain and relevant laws (a lawyer or someone that works in the legal domain). Figure 12 provides a detailed view of the process using BPMN.

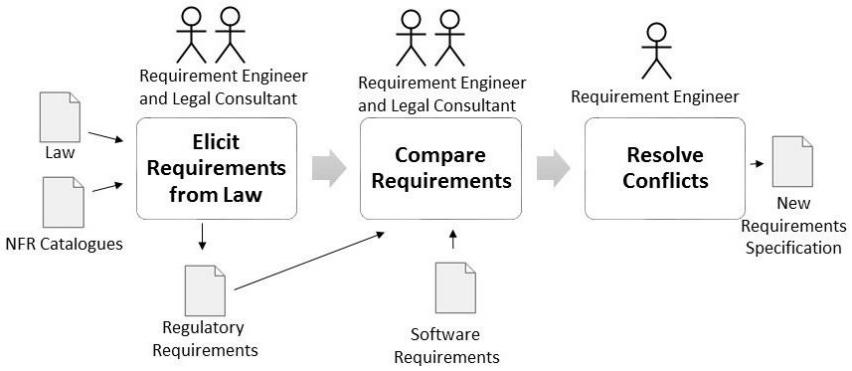

Figure 12 Compliance process

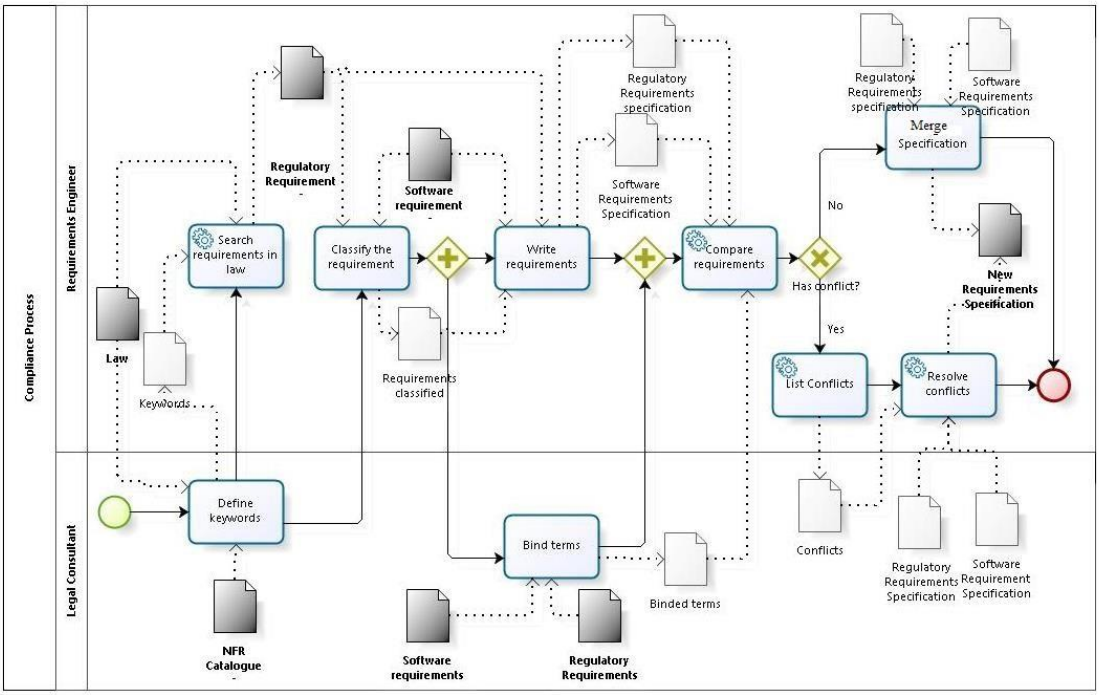

Figure 13 The detailed Process in BPMN

To illustrate our proposed process, we will use the software requirement

"Only authorized users shall have access to students' personal information."

moreover, the law fragment

"The data subject as well as whoever is requested to provide personal data shall be preliminarily informed, either orally or in writing, as to
a) the purposes and modalities of the processing for which the data are intended".

The requirement was reused from an exemplar produced by Masters students at DePaul University for a software system that manages student data for a Nursing School (http://bit.ly/2jG5LUG), and the law fragment is part of the Italian Privacy Law (see http://www.garanteprivacy.it/home en/Italian-legislation).

a. Elicit requirements from Law

When addressing the issue of compliance with a new or updated law, an organization must interpret the text and find the requirements that impact the organization. The first challenge is how to discover in text law the software requirements.

Laws determine people's Rights and Duties. Laws and regulations inform what should be done, what people must do to be compliant (duty or prohibition) and what they are entitled to, but can choose not to (right or permission). Software requirements are also actions that must be performed by the system, so, we use patterns that were already created in NomosT (see Table I) and are the basis for the semantic annotation about right and duties to find the requirements in the laws.

Also, it is necessary to define keywords that are important to the context of the law. If the domain concerns to a quality,– for instance privacy, security, transparency – nonfunctional catalogues (Chung et al. 2000) can help with some keywords that are important in this domain. NFR Catalogues define quality regardingother qualities that help to archive the first one. For example, if we are dealing with privacy to determine the keywordswe can use the NFR Privacy Catalogue (Chung et al. 2000), which will lead us to related keywords: access, openness, disclosure, accountability and protection. These keywords can be complemented by the legal consultant, resulting in more keywords: authorized, data protection, personal data, privacy and private. These keywords plus the definitions of actor, resource, verbs, positive verbs, negative verbs, right and obligation will become the syntactic indicators for the privacy concept in NomosT.

Using our example, NomosT produces the Regulatory Requirements, as per the Italian Law. Figure 14 shows the Regulatory Requirement in XML with the NomosT tags. As such, the requirement is of Right type (by the PositiveVerb), it identifies the Roles (Actor) and the Situation.

```
<Right Id="198" Type="B" Value="Right" IdE="144">
    <Actor Value="data subject" IdE="31309" Id="19" Type="C" />  as well as
    <Actor Value="whoever is requested to provide personal data" IdE="31309" Id="19" Type="C" />
    <PositiveVerb Value="shall" IdE="1156" Id="127" Type="C" /><Action Value="be informed" IdE="3321" Id="60" Type="C" />,
    either orally or in writing, as to:
    <Situation Id="382" Type="B" Value="Situation" IdE="152"><EuList Value="a)" Id="17" Type="S" IdE="23">
    <Resource Value="the purposes and modalities  " IdE="31252" Id="69" Type="C" />
    of <Action Value="processing" IdE="3673" Id="173" Type="C" /><ByConnector Value=" which " IdE="24706" Id="249" Type="C" />
    <Resource Value="the data " IdE="31280" Id="50" Type="C" />
    <Action Value="are intended" IdE="3673" Id="173" Type="C" /></EuList></Situation>
</Right>
```

Figure 14 Regulatory Requirement in XML

It should be noted that the tool helps to annotate and map only what is of interest for the software domain. For this, the presence of legal consultants is critical to guide the search for regulatory requirements.

b.
Compare requirements

To facilitate the comparison between regulatory and software requirements it is necessary to express software and regulatory requirements in the same representation, using the same vocabulary. As indicated earlier, for this we use Desiree.

To do so, we must first classify each requirement according to the Desiree ontology. Following our example, the software requirement "Only authorized users shall have access to students' personal information" is classified as function constraint, and will be represented as FC: Access <object: students' personal information ><actor: only authorized user >.

The regulatory requirement "The data subject as well as whoever is requested to provide personal data shall be preliminarily informed, either orally or in writing, as to the purposes and modalities of the processing for which the data are intended".

(Table 2) is classified as function and state constraint. Trying to facilitate the transformation between NomosT and Desiree, a mapping between Desiree and Nomos Language have been created (Table 2), they are:

Table 2 Mapping from NomosT to Desiree

NomosT	Desiree
Action	verb
Subject	role
Resource	object
Actor	target

Therefore, an Action in NomosT will be represented as a verb in Desiree; a Subject will become a role; a Resource will become an object and an Actor will become a target. In the example, the Actor Value "data subject" will become a target, as well as the Actor Value "whoever is requested to provide personal data." The Action Value "be informed" will be mapped as a verb, and the resource value "the purpose and modalities" will be mapped as an object. As such, the resulting Desiree coding will be: FC:= Inform<object: information><target: data subject>, FC:= Inform<object: information><target: whoever is requested to provide personal data> and SC := Information: << Purpose: String> <Modality: String>.

The legal consultant helpsto map the vocabulary of the law to the vocabulary of the application (Bind terms), this is needed to better anchor the comparison process of the regulatory requirements specification and the software requirements specification. Table 3 shows a partial view of the binding terms from the law to the software requirements for managing a Nursing School student data.

Table 3 Example of binded Terms

Law	Software
Data Controller	Nursing staff, Program Administrators

Data Processor, Persons in charge of the processing	Authorized users, Dr. Susan Poslusny, Dr. Julie Donatek
Data Subject	Students
Personal data, data	Personal information, private information, students grade and presence
Communicate, disseminate, reproduce,	Display
Update, erase, rectification	Modify
Assigned	Transfer

The process compares each requirement in the regulatory requirements by scanning the list of software requirements to see if an existing requirement already operationalizes the regulatory requirement. The regulatory requirements may be:

(a) A conformed requirement: already operationalized in the existing set of software requirements;

(b) A deviated requirement: already partially operationalized in further refinement;

(c) New requirement: not yet operationalized, to be dealt with, or

(d) In conflict with existing requirements.

In cases (b) and (d), requirements engineers should discuss with stakeholders and legal consultants to decide the best way to achieve compliance.

To try to facilitate the comparison we create some guidelines on Table 4:

Table 4 Guidelines to classify the requirements

Type	**guideline**
New requirement	(new requirement) ::= if (new verb)
Conformed requirement	(conformed requirement) :: = if ((same role) + (same verb)+ (same object)
Conflict requirement	(conflict requirement) :: if ((same role) + (opposite verb) + (same object))
	(conflict requirement):: if ((same role) +(same verb)+ (different object))

So, if the regulatory requirementhas a new verb for each, the software requirement does not have a correspondent verb, this indicates a new requirement. If we have the same role, the same verb and the same object in the regulatory requirement and the software requirement it shows a conformed requirement.

For a conflicting requirement, we need to have the same role, the same object and an opposite verb, for instance: if in the regulatory requirement we have the verb keep,in the software requirement we have the verb destroy. On the other hand, if we have the same role, the same verb but different objects, such as: keep information for fiveyears or keep information as long as we need.

In some situations, we will have a non-trivial case, demanding a more in-depthanalysis. For example, if we have the same verbs and the same objects but different roles, we need to verify if there is the same constraint about the roles; if there is, then we have the case of a conflicting requirement, and if it is a new requirement.

Although the comparison is a manual process, it is supported and facilitated by the Desiree framework, as both requirements are represented in the Desiree goal model, which merges the two sets of requirements (Merge Specification). The merged model allows for the tagging of conflicting issues and supports their resolution (Figure 15 Example in Desiree Notation).

Table 5 shows a partial description of this comparison for the example we have been using.

Table 5 Comparison between regulatory and software requirements

Regulatory Requirements	**Software Requirements**	**Comparison**
	FC: Access <object: students' personal information ><actor: ONLY authorized user >.	
F1:= Inform<object: information><target: data subject>,		New Requirement
F2 := Inform<object: information><target: whoever is requested to provide personal data>		New Requirement
SC1 :=Information: << Purpose: String> <Modality: String>.		New Requirement

Table 5, on the other hand, indicates yet another regulatory requirements for which a conflict arises.

The Software requirements are:

- The system shall be able to display the student's names and grades
- The system shall be able to display the student's frequency
- The system shall be able to display a printable summary for individual nursing students, which will include (but not be limited to) student name, student ID, admission date, classes, credits, GPA and the cohort that the student is enrolled in.

- The system shall be able to keep the data for 5 years

The Regulatory requirements are:

- "It shall be prohibited to communicate and disseminate personal data for purposes other than those specified in the notification as per article 7."

- "Personal data undergoing processing shall be kept and controlled, also in consi deration of technological innovations, of their nature and the specific characteristics of the processing, in such a way as to limit to the very minimum, by means of suitable security measures, the risk of their destruction or loss, even if accidental, of unauthori zed access to the data or of their being processed unlawfully or in a way that is not consiste nt with the purposes for which they have been collected."

- "Data may be:a) destroyed;"

- "The data shall be reproduced on paper or magnetic media, or else transmitted via electronic networks"

- "The system shall be able to not use any personal data that is processed in breach of the relevant provision concerning the processing of personal data"

The regulatory requirement is processed by Nomos T, both sets of requirements are put in Desiree language and after comparison to find the possible conflicts:

Table 6 An Example of Comparison with Conflict

Regulatory Requirements	Software Requirements	Comparison
F:Responsible<subject: data processor><object: retrieval of data>		New Requirement
FC:= Not Communicate <subject:{system}><object: personal data> FC:= Not Dissiminate<subject:{system}>	FC:= Display <subject:{system}><objetct: students names and grades > FC:= Display <subject: system><objetct: frequency>	Conflicting Requirement
QC:= unauthorizedAcessPrevention(studentRecord): safe	FC:= Display <subject:{system}><objetct: students names and grades > FC:= Display <subject: system><objetct: frequency>	Conflicting Requirement
FC: Not use <subject: {system}><object: personal data><except: that is process personal data>		New Requirement
F:=Reproduce <subject:{system}><object: data><means:paper> F:= Reproduce <subject:system><object: data><means: magnetic media or transmitted via electronic networks >	F:= Display<subject:{system}><object: printable summary for individual nursing students> SC := printable summary:<<Student id: String><name: String><admission Data: Date><classes credits: Integer><GPA: String>< cohort that the student is enrolled in : String>;	Conformed requirement
F:= Destroy<subject: system><object:data>	F:= Kept <subject: system><object:data><when : for 5 years >	Conflicting Requirement

Using the tool (Figure 15 Example in Desiree Notation), we can mark the conflicts, by pointing to possible resolutions: dropping, strengthening or weakening the conflicting requirements. The tool also supports the operationalization of possible decisions.

c. Resolve conflict

To resolve conflicts we have three possible strategies:

1- Weaken the software requirement: by transforming a complex requirement (a requirement with multiple concerns) into atomic ones (a requirement with a single matter). For example: “The system shall collect personal info in real time” to “The system

shall collect personal info.” The first one can be fulfilledif the collect of personal info was in real time; the second one can be fulfilled for any solution thatcollects personal data;

We will use the strategy of week software requirement when conflict arises because the software requirement has more detail information that conflicts with the regulatory information.

2- Strengthen thesoftware requirement: here we transform atomic requirement into a complex requirement (adding more concerns). For example *“The system shall book tickets”* to *“The system shall book an online ticket.”* The first one can be fulfilled by any solution thatcan book a ticket, but the second one only can be fulfilled if a solution canbook an airfare ticket (more specific one);

In this case, the conflicts arise because some information is missing in software requirement, so we strengthen the software requirement adding more details.

3-Drop the software requirement (in the last case).

We need to analyze both conflicting requirements, always remembering that regulatory requirements have higher priority over software requirements.

Figure 15 Example in Desiree Notation, shows how the case of Table 6 is modeled as a Desiree goal model as to support Conflict Resolution.

For example:

- Software requirement FC:= Display <subject:system><object students names and grades >
- Regulatory requirement: FC:= Not Communicate <subject:system><objetc: personal data>

The above example shows a conflict that can be resolved by strengthening the software requirement putting a constraint that the personal data can only be accessed by an authorized user so thatthe new requirement will be:

FC:= Display <subject:system><object students´ names and grades ><target: only authorized person>.

On the other hand, suppose the following situation:

Regulatory requirement FC:= Destroy <subject: system><object:data><when: when data processing be terminated>

Software requirement F:= Destroy <subject: system><object:data><when: after 5 years>.

These requirements have conflicting effects, so the only way to resolve the conflict is to drop one of them.Once the regulatory requirement needs to be satisfied we drop the software one. The result of the process is a specification that includes new regulatory requirements, while some software requirements have been dropped, others have been modified (Figure 15).

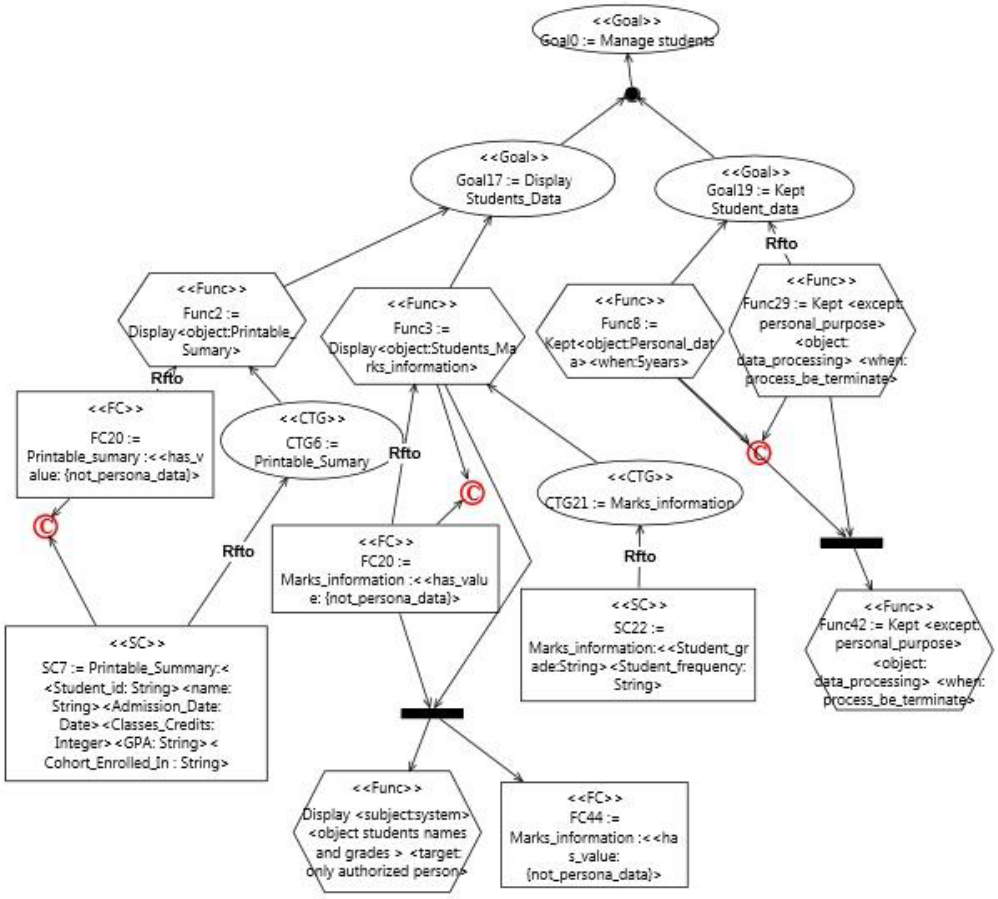

Figure 15 Example in Desiree Notation

Figure 15 shows an example of Desiree tree with the Conflict Resolution.A requirement specification to manage student school software is shown,and 3 conflicts marked with a red C areshown. So the goal is to manage students. for this, it is necessary to Keep Student Data (goal 17) and Display Student Data (goal 19). In the privacy law is written that personal data cannot be displayed (FC: Marks-information <has_value: { not_personal_data}> and in the software requirement that the student grades need to be displayed (Func 3 = Display<object Students_marks_information>). So we have a conflict, marked as C, to resolve this conflict we strength the software requirement putting that the grades will be shown only to the authorized person, as showed after the black line Func: Display <subject:system><oject:students name and grades><target:only authorized person> .

4.3. Compliance Monitoring

A compliance process elicits requirements from the law, compare them to existing software requirements and resolve possible conflicts, resulting in a compliant specification (Engiel, Leite & Mylopoulos, 2017). The specification may include new compliance requirements, while some software requirements may have been dropped, others have been modified.

Laws, requirements and the environment where the software operates can change, and the software needs to stay compliant. For this reason, it is necessary to monitor the

changes and act to the software remains legally compliant. A system is an organized structure that consists of a set of elements interrelated and interdependant. The system has inputs, outputs, andfeedback mechanisms to achieve one goal despite a change in the external environment. The input for the system is the Law and the SRS, the output of the system is the SRS compliant. Changes occur,and the feedback loop needs to promote that the SRS remains compliant (Figure 16).

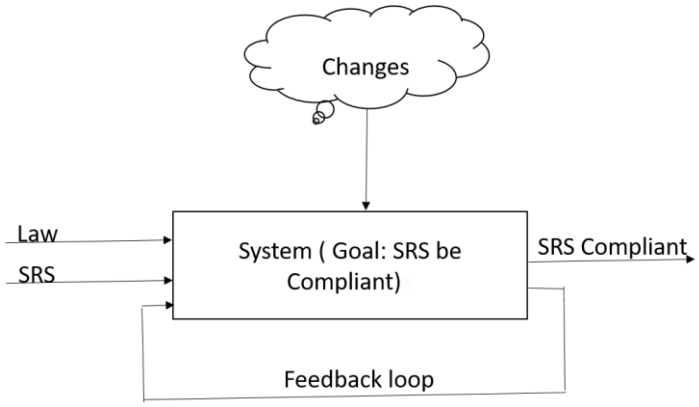

Figure 16 - System that supports compliance

Feedback loops help the software to self-adapt (Silva Souzaet al. 2011). It focuses on the activities that realize feedback: monitoring, analysis, plan, execution – MAPE (Kephart, Chess, 2003). A feedback loop would: (1) monitor data that are important to the stakeholders; (2) compare the monitored data with previous ones; (3) if the monitored data do not match with the expected one, a solution needs to arise. Therefore, the proposal uses a feedback loop to be aware of the context change to promote that the SRS remains compliant.

The MAPE is represented as agents because the changes in the environmentare not predictable, they can happen very fast and the system needs to be capable of deciding what to do in all situations to achieve their goals. Agents are entities that are continuous active because of their observation of the environment, which needs to modify their intern status according to the environment to perform actions. Agents have the autonomy to act independently of human activity, they can take decisions analyzing the situations and their experience, as they can also react to external stimulus and work with other agents to achieve one common goal.

We have three cases of changes: change in the law, change in the requirement and change in the context that will have impacts on how MAPE will work.

4.4. Law changes

The first case is when laws are updated, or new laws arise. Organizations must regularly check for new laws relevant to their businesses. For example, in the UK, people can register their email and the system sends the information about the laws that the people are interested in (http://www.legislationupdateservice.co.uk/). In Brazil, every month, all the updates are publicized in the Official Gazette, in order to people interested can search for a word on this page every month.

After comparing new legislation with the one previously in use, organizations need methods to identify which parts of the system or steps of procedures should be modified to make the system compliant with the new laws. When a change is verified, it is necessary to start the compliance process: discover the new regulatory requirements in the law, compare it with software requirements and resolve possible conflicts. Figure 16 shows the case, with the significant activities to deal with.

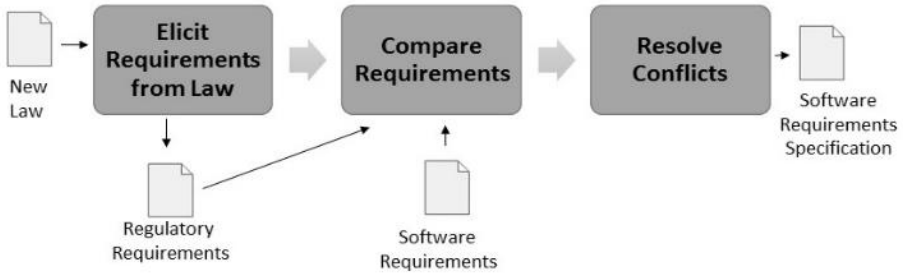

Figure 17Case of legislation changes

In this case, we have the Software Requirement (SR) and the law or a set of laws (L) that need to be compared to promote compliance (C). So we can say to encourage Compliance is a combination of software requirement (SR) and a set {} of laws (L) represented as C = SR AND {L}.

We can represent the case in a SR Model (Figure 18), to clarify how the feedback loop promotes the continuous compliance. Since SR Models allows modeling of the reasons associated with each agent and their dependencies and provides information about how actors achieve their goals and soft goals. We have fouragents on the monitor, the analyzer, the planning and the executor, one role the legal office and one actor the software (Figure 18).

The Monitor, Executor, Analyzer and Planning agents represent people since the MAPE will occur outside of the Software. The Monitor, Planning, Analyzer and Executor agent is a person, an actor (represented by an ellipse with a line in the top). The Monitor agent is responsible for monitoring the changes in legislation and can play the role (represented by an ellipse with a line in the bottom) of Legal Office, i* Strategic Actor model (Figure 18) represents the fact that a person, as a monitoring agent, may play the role of the legal office.

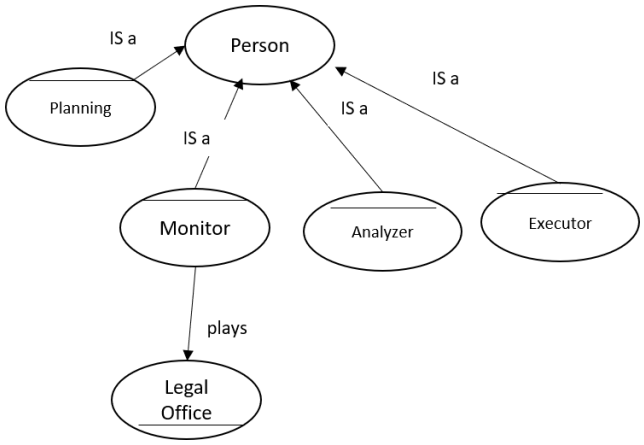

Figure 18 SA Diagram for Legislation Changes

The Legal Office is an external actor that is responsible for the changes (update or create new laws). The Monitor, Analyzer, Planning and Executor (MAPE) are internal agents of the system (but external of the software), who are responsible for keeping the SRS compliance. The monitor agent aims to verify the legislation to promote that the changes will be found. When a change is identified, it is necessary to prepare the system for the changes. To do this is required to analyze the impact in SRS, make a plan for the changes and then execute the plan: perform compliance to generate the new software requirement specification (SRS). The Executor is responsiblefor running the compliance process and creating the new SRS.

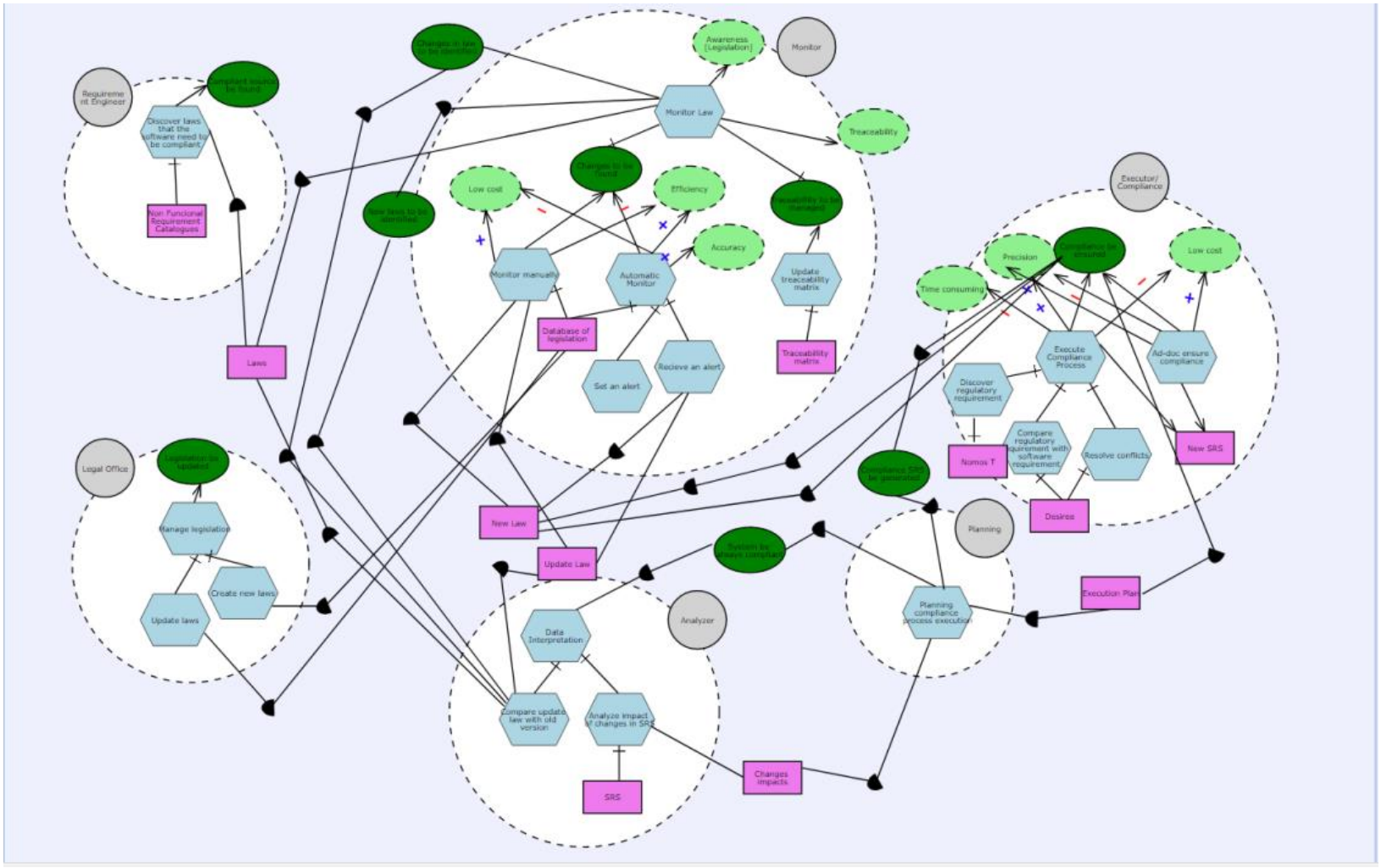

Figure 19 SD for the case of changes in legislation

To define how each task (which is a leaf in the diagram) will work it was used the BPMN diagram (Business Process Management Notation) (OMG, 2011) and the Desiree Software Specification (Li et al. 2015). For each task, one subsection was created and the traceability to SR diagram can be done using the string of the task name.

a.
Requirement engineer agent:

The requirement engineer agent aims to achieve the goal of Compliance source be found. For this, he executes the task of discovering laws that the software needs to be compliant (Figure 20). For this task, the agent uses the Non-Functional Requirements Catalogues.

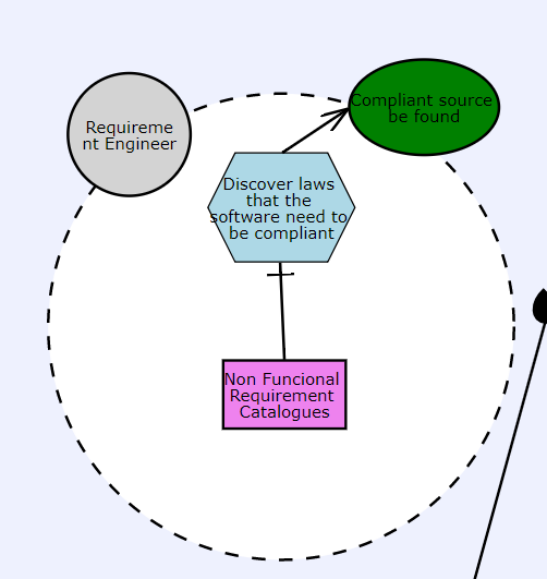

Figure 20 Requirement engineer agent

- **Discover laws that the software need to be compliant with**

In Figure 21 the BPMN diagram for Discover laws that the software need to be compliant with is showed. First, it is necessary to discover what laws the software needs to be compliant with. If it is a non-functional law, a requirement engineer can use the Non-Functional Catalogues to define keywords. If it is a specific domain it is necessary to consult a law specialist about the laws and keywords.

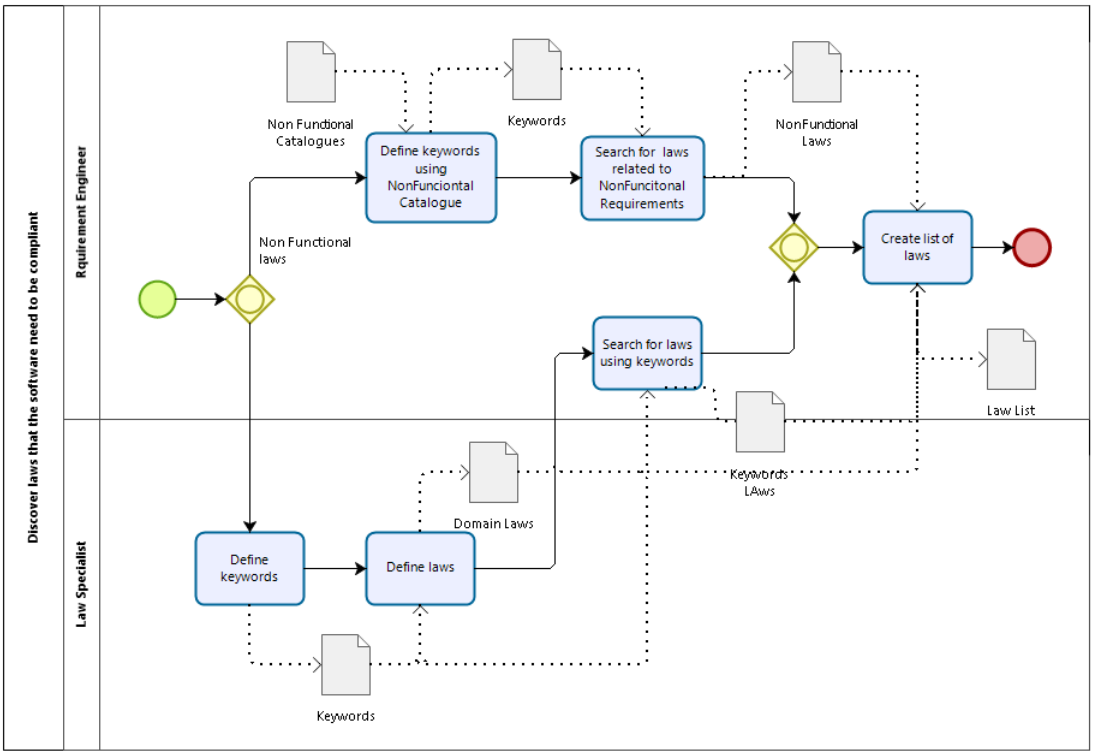

Figure 21Discover laws that the software need to be compliant with

|The agents were specified in Desiree Language to have a requirement specification that can be used to develop the agents. In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need six functions requirements that are represented as Function (F) in Desiree:

G: List of laws to be created

F: Define <object:keywords><subject:law_specialist>

F: Define <object: keywords_non_functional><subject:requirement_engineer ><parameter:non_functional_catalogues>

F: Define <object: laws><subject:law_specialist>

F:Search <object:law><subject:requirement_engineer><where:law_database><parameter:keyword>

F:Search <object:law><subject:requirement_engineer><where:law_database><parameter:NonFunctionalkeyword>

F: Create <object: List_laws><subject:requirement_engineer>

b.
Monitor agent:

The monitor agent aims to achieve the goal of changes in the law to be identified. For this, he executes the task of monitor law, that can be done in two ways, monitor manually, that contributes positively for the low cost but negatively for efficiency or automatic monitor that contributes negatively for the low cost but positively for efficiency and accuracy. For this task, the agent uses a database of legislation. When the monitor law also needs to update the traceability matrix, to achieve the goal of traceability, so it is managed to update the traceability matrix (Figure 5).

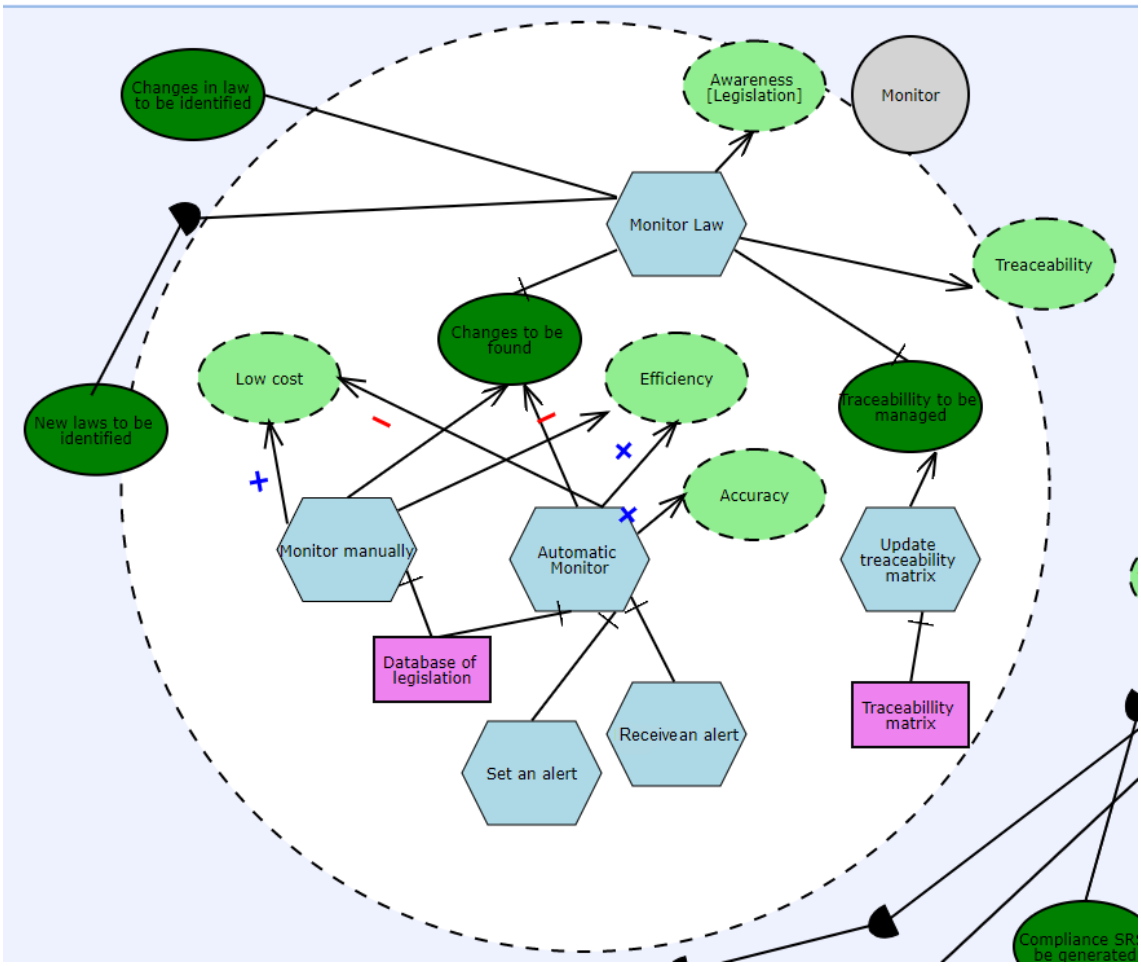

Figure 5 Monitor agent

- **Monitor manually**

In Figure 9 the BPMN diagram for monitor manually is showed. To monitor the changes manually it is necessary to enter into law database and search for the laws using the keywords defined. If a new law or an updated law is found, it is needed to download the latest version.

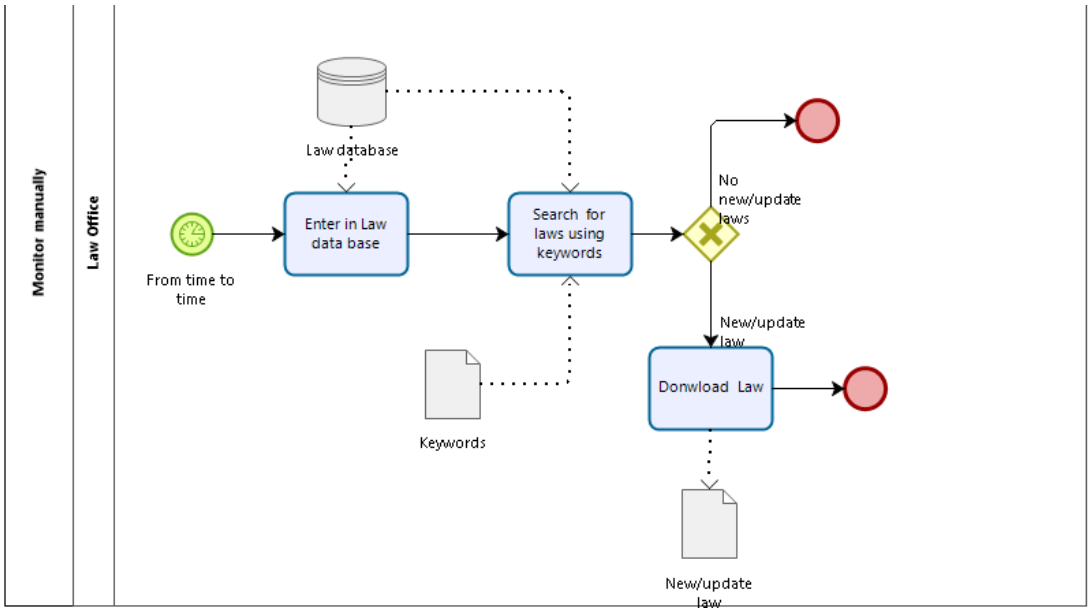

Figure 6 Monitor manually

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal, we need three functions requirements that are represented as Function (F) in Desiree:

G: Changes in law to be found

F: Enter <object:law_database><subject:law_specialist>

F:Search <object:update_law><where:law_database><subject:requirement_engineer><parameter :keyword>

F: Download<object:law>

- **Set an alert**

The BPM diagram for setting an alert is shown in Figure 10. To monitor automatically it is necessary to set the alarm for all the laws that we are interested and can affect the system.

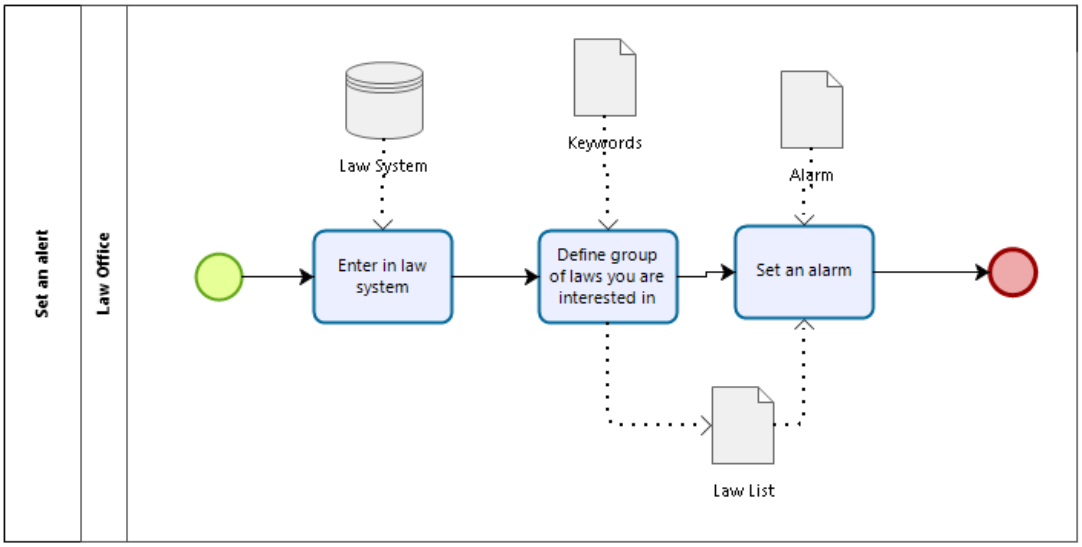

Figure 7 Set an alarm

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need three function requirements that are represented as Function (F) in Desiree:
G: Alert to be set

F: Enter <where: law_system><subject:law_specialist>

F: Define<object:group_of_law><subject:law_specialist>

F:Set

<object:alarm><subject:law_specialist><where:lawDatabase><parameter:ListofLaws>

- **Receive an alert**

In Figure the BPM diagram for receiving an alert is shown. When an updated law or a new law is found automatically, an alert is received. This alert can arrive by e-mail or as a notification in asmartphone. After receiving the alert is necessary to enter into the system and download the new version or the new law.

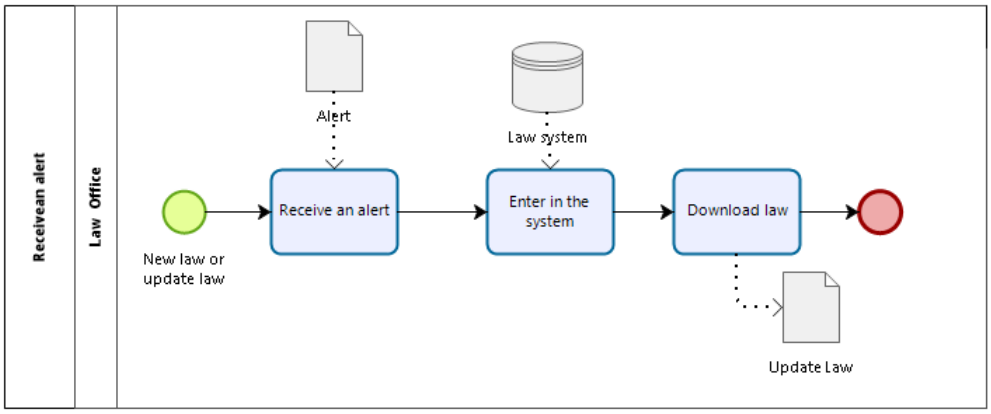

Figure 8Receive an alert

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need three functions requirements that are represented as Function (F) in Desiree:
G: Changes be to alert

F:Receive <object:alert><subject:law_specialist >

F:Enter <where:system><subject:law_specialist>

F:Download<object:law><subject: law_specialist >

- **Updatetraceability Matrix**

The traceability matrix maintains all the traces, relating the software requirements with the regulatory requirements. So is necessary to update the traceability matrix with the updates in law and in the requirement specifications. Figure 26shows the BPMN diagram.

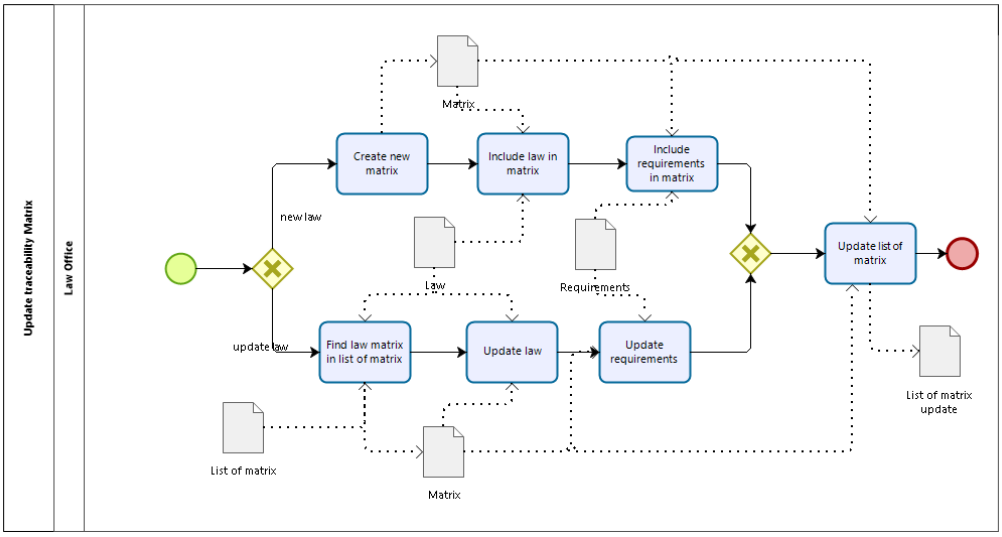

Figure 26 Update traceability matrix

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal, to achieve these goals we need seven function requirements that are represented by Function (F) in Desiree:

G: Trace to be updated

F: Create <object:matrix><subject: Requirement_engineer >

F: Include <object:law><parameter:matrix>><subject: Requirement_engineer >

F: Include <object:requirement ><parameter:matrix>><subject: Requirement_engineer >

F: Find <objec:matrix><parameter:list_of_matrix><subject: Requirement_engineer >

F: Update <object:matrix ><parameter:law ><subject: Requirement_engineer >

F: Update <object:matrix><parameter:requirements ><subject: Requirement_engineer >

F: Update <object: matrix><subject: Requirement_engineer ><when:changes_found>

c.

Legal Office Agent:

The legal office agent aims to achieve the goal of legislation to be updated. For this, he executes the task of managing legislation that can be the task of creating a new law or update law (Figure 27).

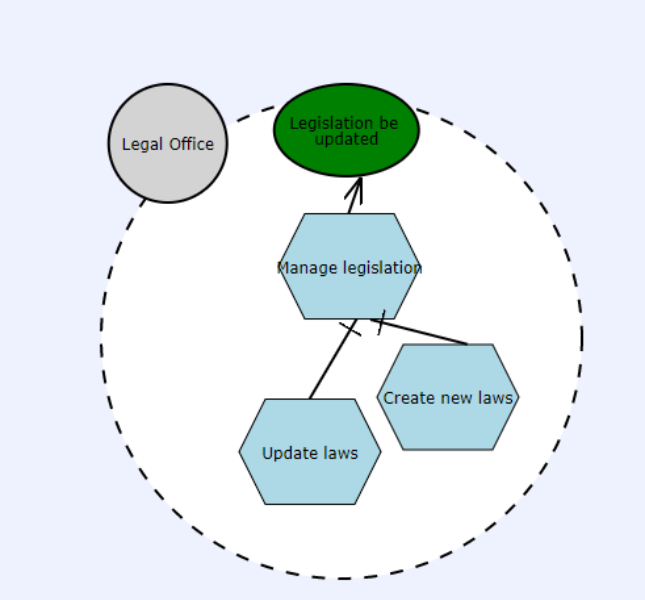

Figure 27 - Legal Office agent

- **Update Law**

Figure 28shows the BPMN diagram. When it is necessary, the Legal Office can create a new version of a law to update paragraph or to create new rules. For this, first it is necessary to find the law in the database, create the changes (that are already approved), update the law text and add the updated version in the database.

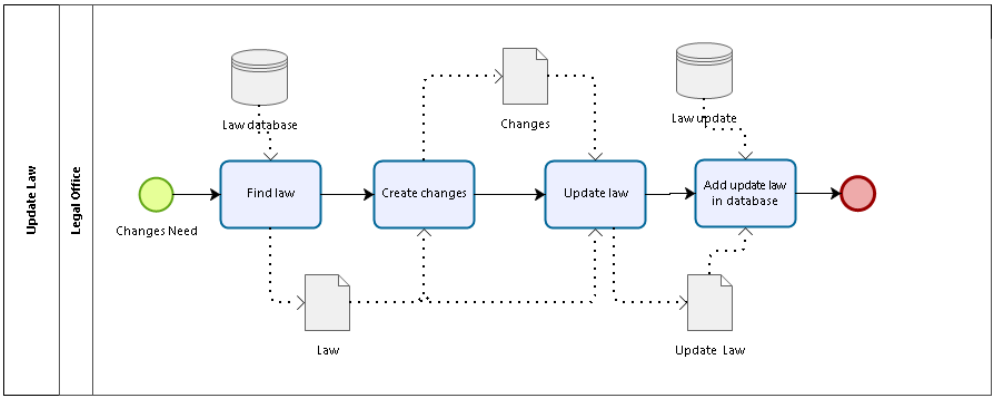

Figure 28 Update Law

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal,to achieve this goal we need four functions requirements that are represented by Function (F) in Desiree:

G:Legislationto be updated

F: Find <legislation><subject:legal_office><where:law_database>

F: Create <object:changes><subject:legal_office><parameter:law>

F:Update <object:legislation><subject:legal_office>

F: Add<object:update_law><subject:legal_office><where:law_database>

- **Create new Law**

When is necessary, the Legal Office can create a law. For this, first is required to create the new law (that are already approved) and add the updated version to the database. Figure 29 shows the process represented in BPMN.

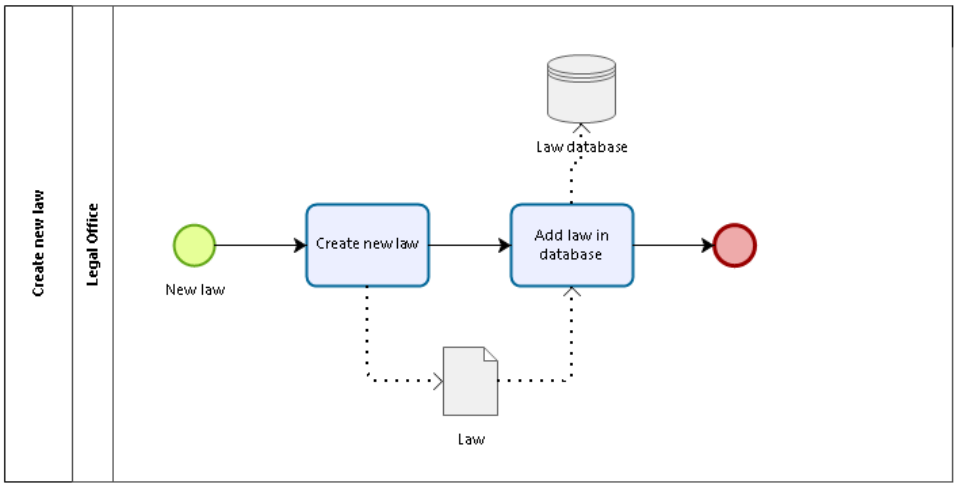

Figure 29Create a law

In Desiree Language, the task will be decomposed in a goal (G), to achieve this goal, to achieve this goal we need two functions requirements that are representedby Function (F) in Desiree:

G:Legislation to be updated

F:Create <object:law><subject:legal_office>

F: Add<object:law><subject:legal_office><where:law_database>

d. Analyzer agent

The analyzer agent aims to achieve the goal of the system to be always compliant. For this, he executes the task of data interpretation that is decomposed in Comparing update law with the old version and Analyzing the impact of changes in SRS (Figure 30).

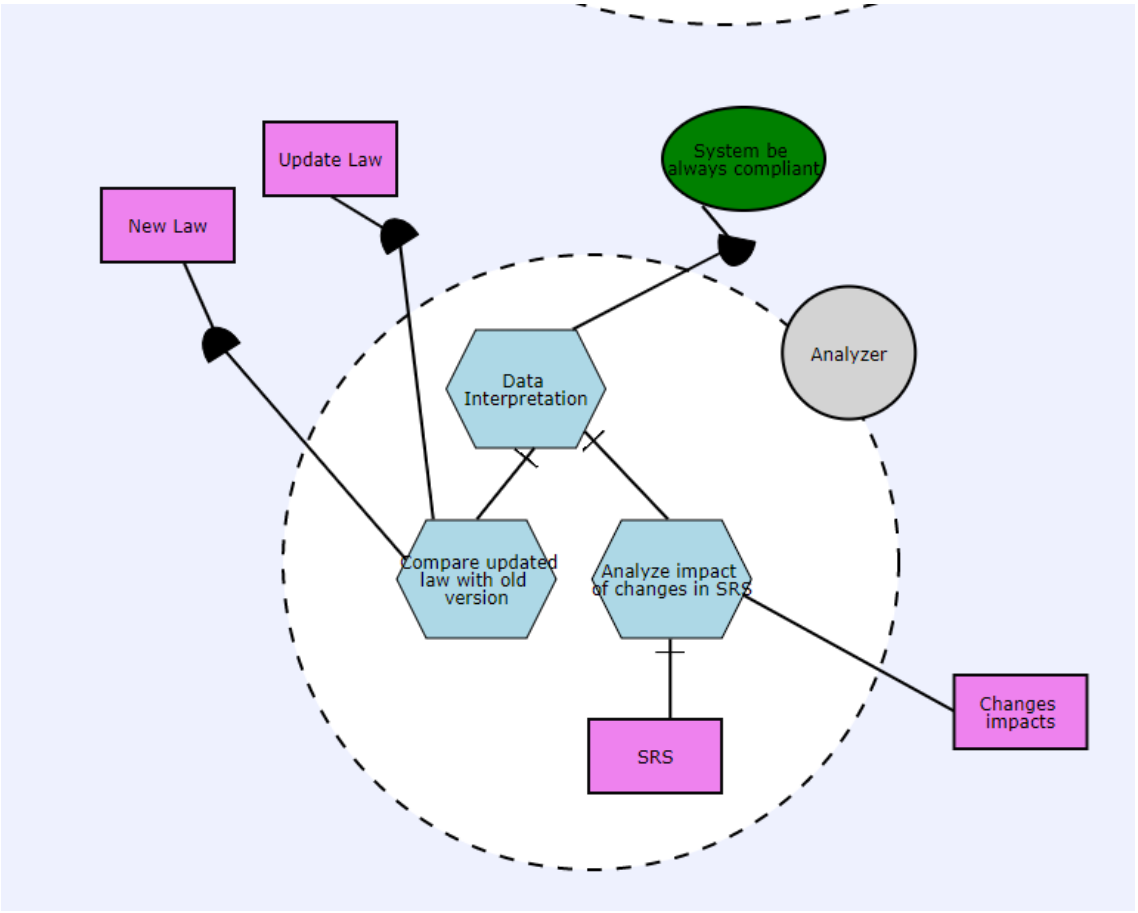

Figure 30 Analyzer agent

- **Compare law with old version**

To analyze the changes, it is necessary to read the updated version and compare it with the old version. If only some sections are updated it is necessary to highlight those changes. Figure 31 shows the BPMN describing the process.

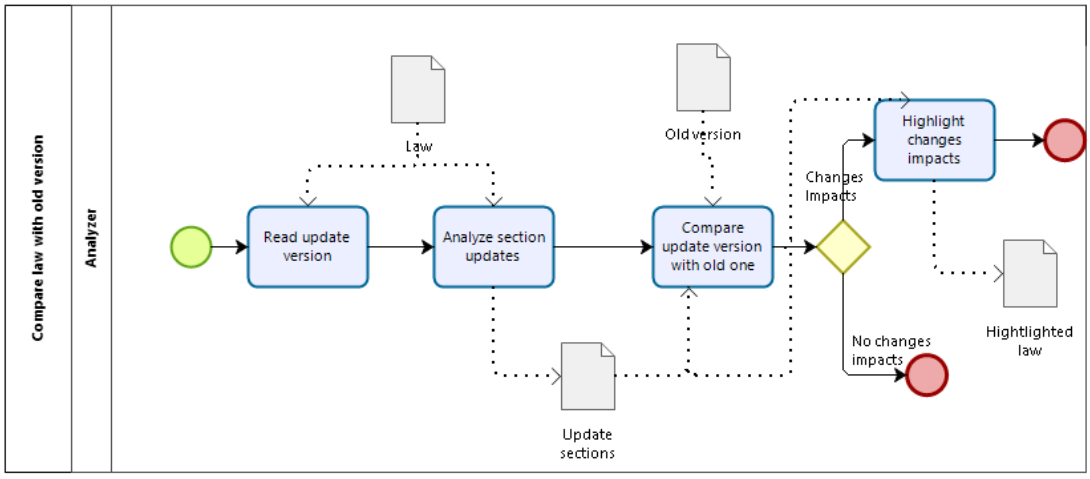

Figure 31 Compare law with old version

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need 4 functions requirements that are represented by Function (F) in Desiree:

G:Changes in law to be identified

F:Read<object:update_version><subject:Requirement_engineer>

F:Analyze<object:
sections_update><subject:Requirement_engineer><parameter:update_version>

F:Compare
<object:old_version,law_text,update_version><subject:Requirement_engineer>

F: Highlight<object:
changes><subject:Requirement_engineer><parameter:update_version>

- **Analyze changes impacts in SRS**

It is necessary to analyze if the law changes the impact on the software requirement specifications. For this, it is necessary to verify if the modifications parts have regulatory requirements, then if these regulatory requirements have software requirements associated. Figure 32 shows the BPMN diagram

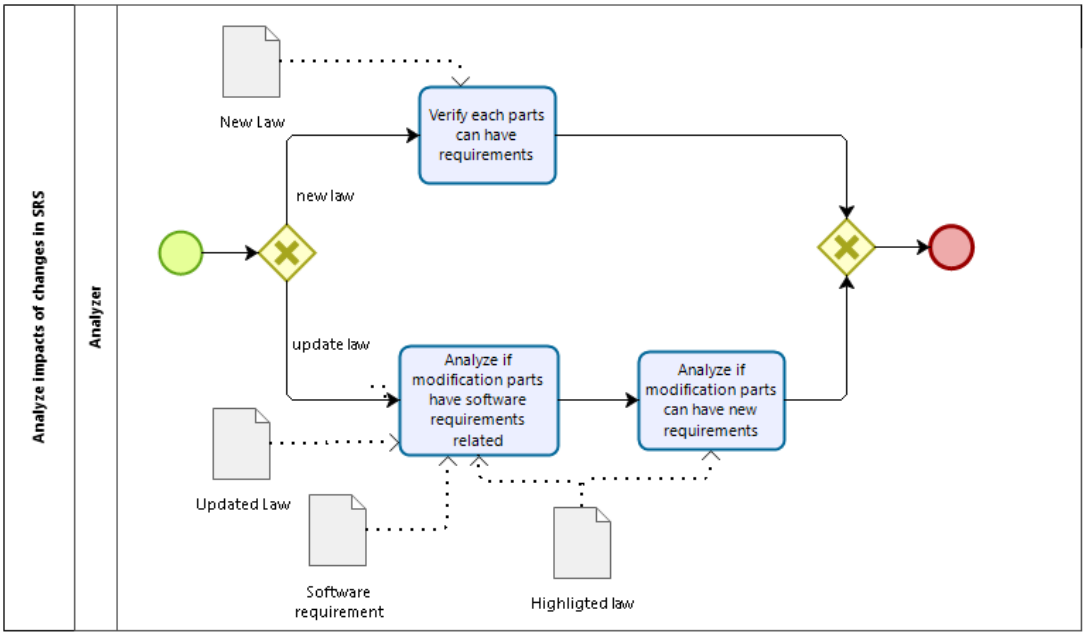

Figure 32 Analyze changes impact in SRS

In Desiree Language we will have a goal (G) that is the same of the i*, to achieve the goal we need three functions requirements that are represented by Function (F) in Desiree:

G:Change Impacts to be analyzed

F: Verify<object:new-law><subject: Requirement_engineer >

F: Analyze <object:highlighted_law ><subject:Requirement_engineer ><parameter: Software_requirements>

F: Analyze <object:highlighted_law ><subject:Requirement_engineer ><parameter:new_requirements>

e. Planning agent

The planning agent aims to achieve the goal of Compliance SRS to be generated . For this, he executes the task of Planning Compliance Process Execution (Figure 33).

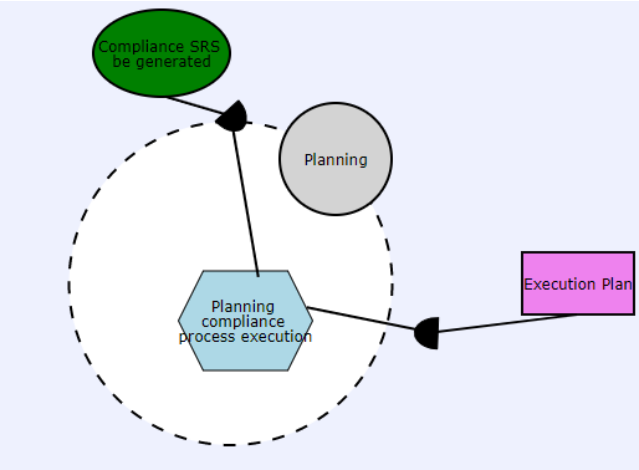

Figure 33- Planning agent

- **Planning compliance process execution**

It is necessary to plan how the compliance process will be executed: using the entire law, part of the law, or maybe it will be not necessary to run the compliance process if the changes do not affect the requirements.

Figure 34 shows the BPMN diagram.

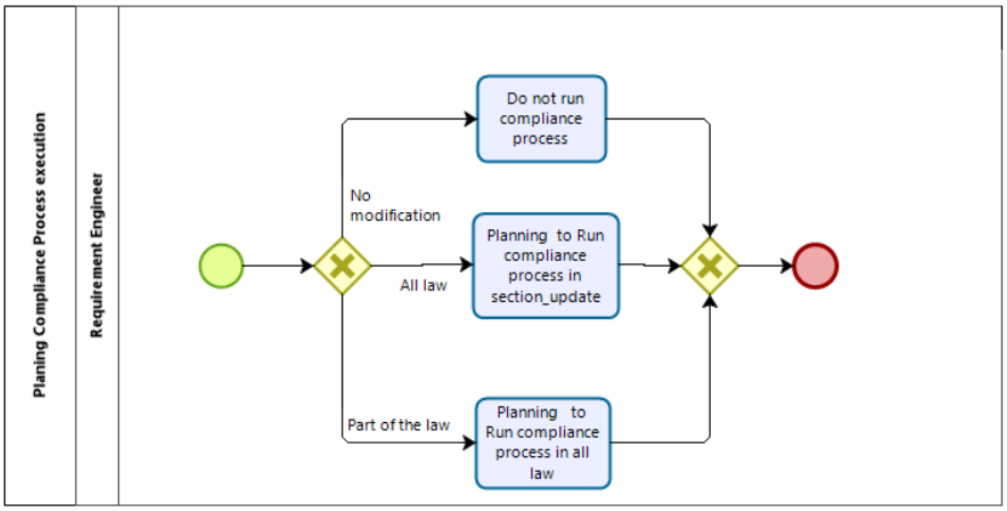

Figure 34 Planning Compliance Process Execution

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal, we need three functions requirements that are represented by Function (F) in Desiree:

G:Compliant Process to be planned

F:Plan<object:complianceProcess><where:law><subject: Requirement_engineer >

F: Not run<object: complianceProcess><where:law><subject: Requirement_engineer >

F: Plan<object:complianceProcess><where:part_of_law ><subject: Requirement_engineer >

f.
Executor agent

The executor agent aims to achieve the goal of Compliance to be Ensured. For this, he can execute the Compliance Process or an ad-doc method to ensure compliance. And ad-doc is low cost but has less precision and is time-consuming. Because of this he always executes the compliance process that is composed of discovering regulatory requirements, comparing regulatory requirements with software requirements and resolving conflicts (Figure 35).

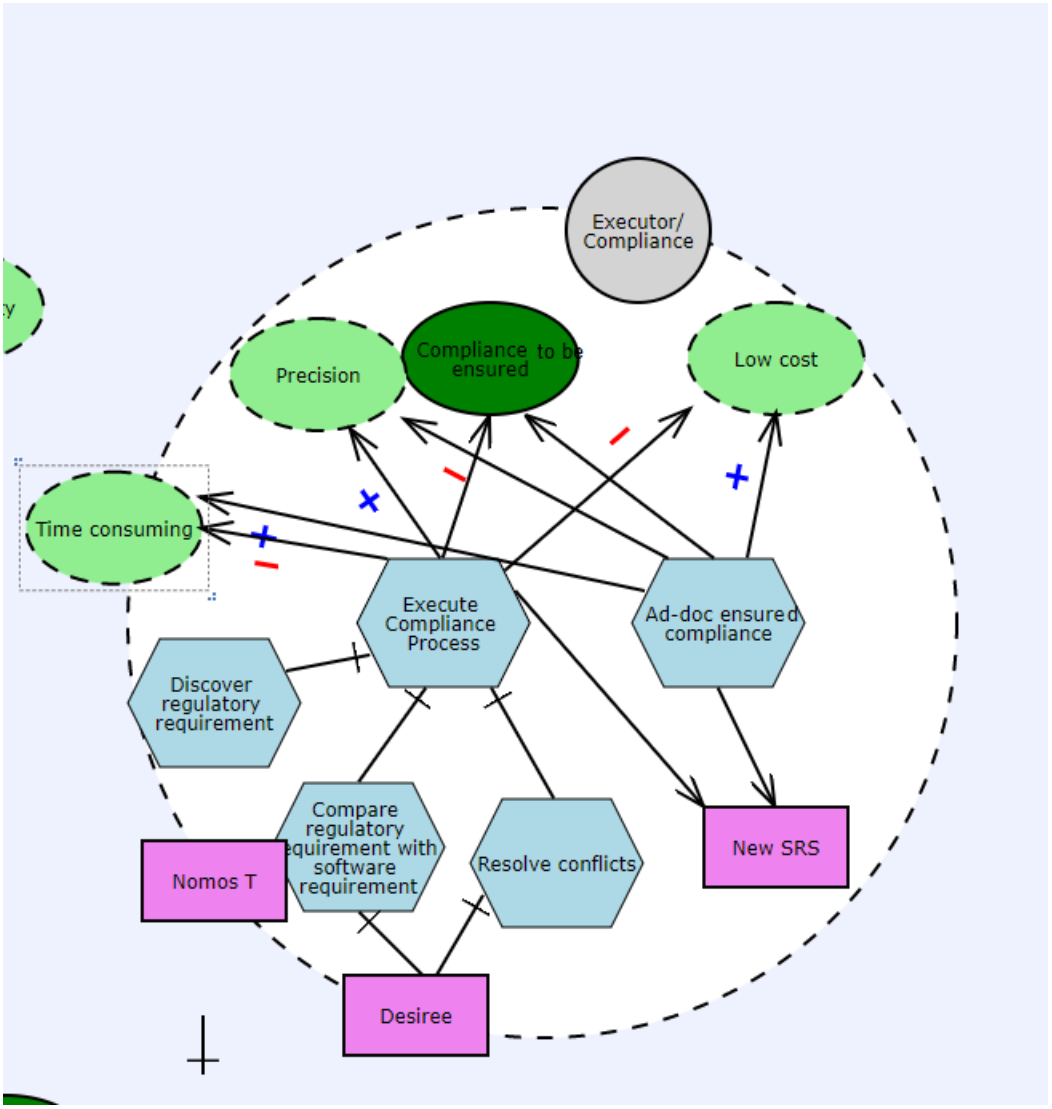

Figure 35 - Compare update version with older version

The tasks related to the Execute Compliance Process (Discover regulatory requirements, Compare regulatory requirement with software requirements, Resolve Conflicts) were already specified in item 4.2.

4.5. Requirements changes

When software requirements change, also the the system needs to remains complaint. As a premise it is assumed that the software organization has ways of tracking requirements changes, using a requirements configuration management system. Therefore, as a change occurs, it is necessary to verify how the changes affect the regulatory requirement: if those changes generate some conflicts with the regulatory requirements (compare requirements) and if it is necessary to resolve possible conflicts (Engiel, Leite & Mylopoulos, 2017).

Figure 36 shows this case, where the activities needed to be executed in gray. It is not necessary to elicit the requirements of the law since they have not changed so that the process can start in the second activity. We compare the new software requirements with the regulatory requirements that already exist to see if some conflicts are added and then resolve the conflicts.

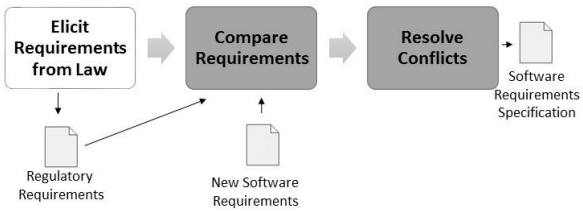

Figure 36 Case of software requirement changes

In this case, we also have the Software Requirement (SR) and the law or a set of laws (L) that need to be compared to promote compliance. So Compliance is a combination of software requirement (SR) and a set {} of laws (L) represented by C = SR AND {L}.

The software requirement changes also are represented as a SR Model (Figure 37), to show how the agents will act. In this case, it hasfour agents, the monitor, the analyzer, the planning and the executor, one role, the requirement engineer, and two actors - the software and the client. The Client asks for the changes, and the requirement engineer will implement it, the client and the requirement engineer are external actors. The Monitor, Analyzer, Planningand Executor (MAPE) are internal agents of the system that are responsible for keeping the SRS compliance. The monitor agents aim to verify changes in requirements. When a change is identified, it is necessary to analyze the impacts in SRS: if it has regulatory requirements associated with them. If it has, it is necessary to make a plan to perform the compliance process to generate the new software requirement specification (SRS). The Executor is responsiblefor running the compliance process to generate the compliance specification.

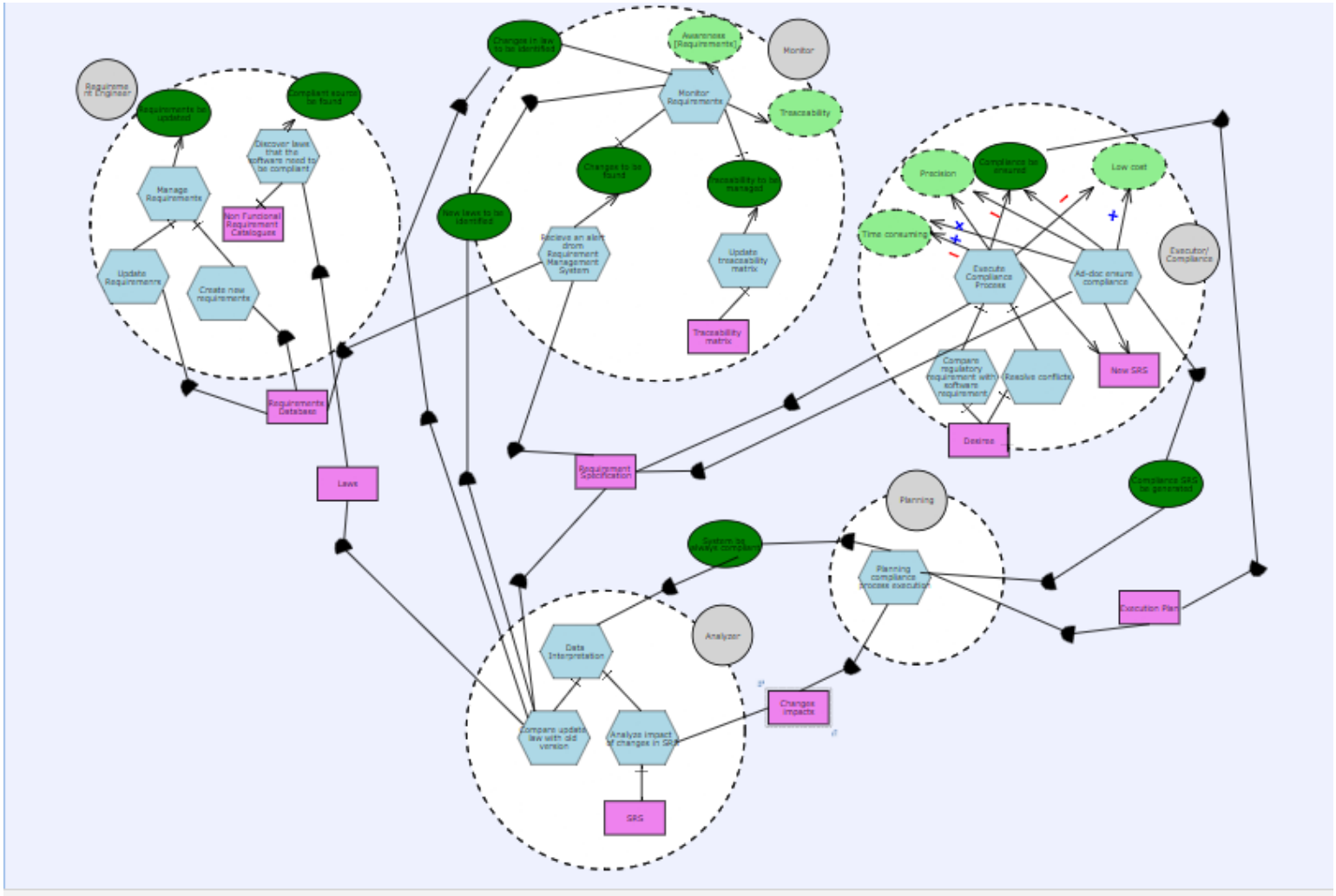

Figure 37 SR for the case of changes in requirement

The Monitor, Executor, Analyzerand Planning agents are persons since the MAPE will occur outside of the Software.

The i* Strategic Actor model (Figure 38) represents the fact that a person, as a monitoring agent, an analyzer agent, a planning agent or an executor agent, may play the role of requirement engineer. The software is an actor that will receive the new SRS to be implemented.

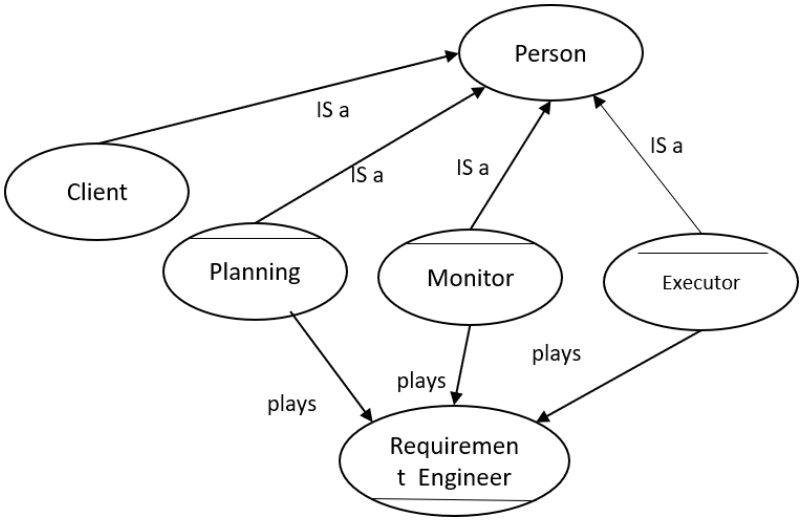

Figure 38 SA diagram for changes in requirements

For each task (which is a leaf in the SR diagram) we would use the BPMN and Desiree as languages to specify the details of each leaf task in the SD diagram. The specification defines the requirements for each task.

a.
Requirement engineer agent

The executor agent aims to achieve the goal of Compliance source to be found, and requirements are updated; For this, he needs to discover laws that the software need to be compliant with and manage requirements that can update the requirements or create new ones if the client asks for changes in software (Figure 39).

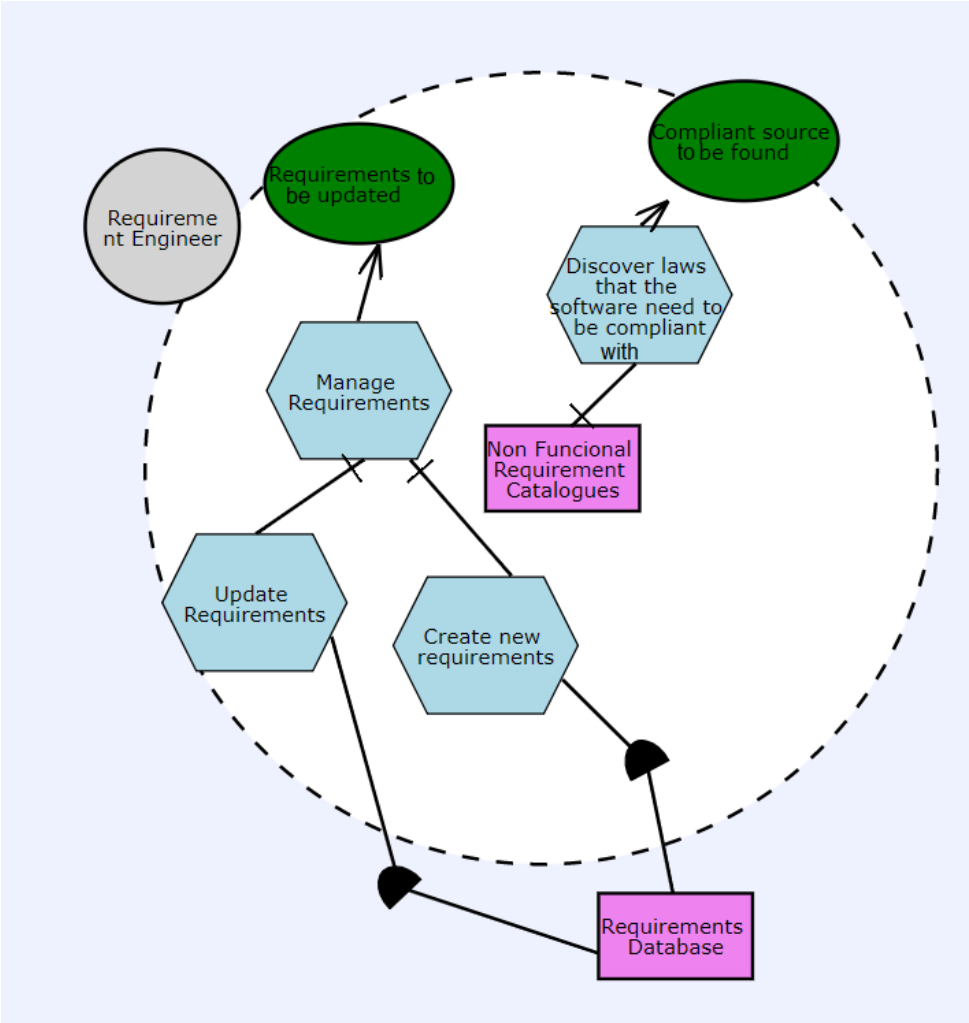

Figure 39 Requirement engineer agent

- **Discover laws that the software need to be compliant with**

The discovering of laws that the software need to be compliant with will occur in the same way that occurs for the changes in law, already represented in Figure 21.

- **Update requirements**

When clients ask for changes in functionalities, it is necessary to update the requirements. Figure 40 shows the BPMN diagram.

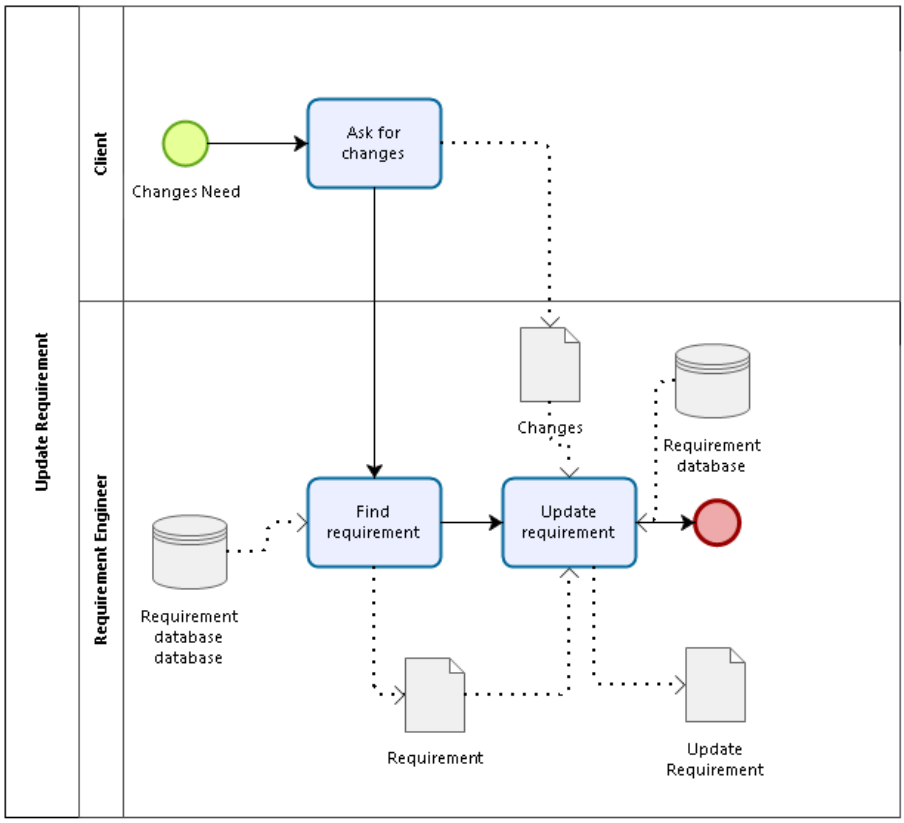

Figure 40 Update requirements

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal is necessary three functions requirements that are represented as Function (F) in Desiree:

G:Requirement to be changed

F:Ask

<object:modifications><subject:cliente><target:requirement_engineer><parameter:requirement>

F: Find <object:requirement><subject:requirement_engineer>

F:Update <object:requirement><subject:requirement_engineer>

- **Create new requirement**

When a client asks for new functionality, it is necessary to create a new requirement. After the requirement is created, it is necessary to include it in requirement database. Figure 41 shows the BPMN diagram.

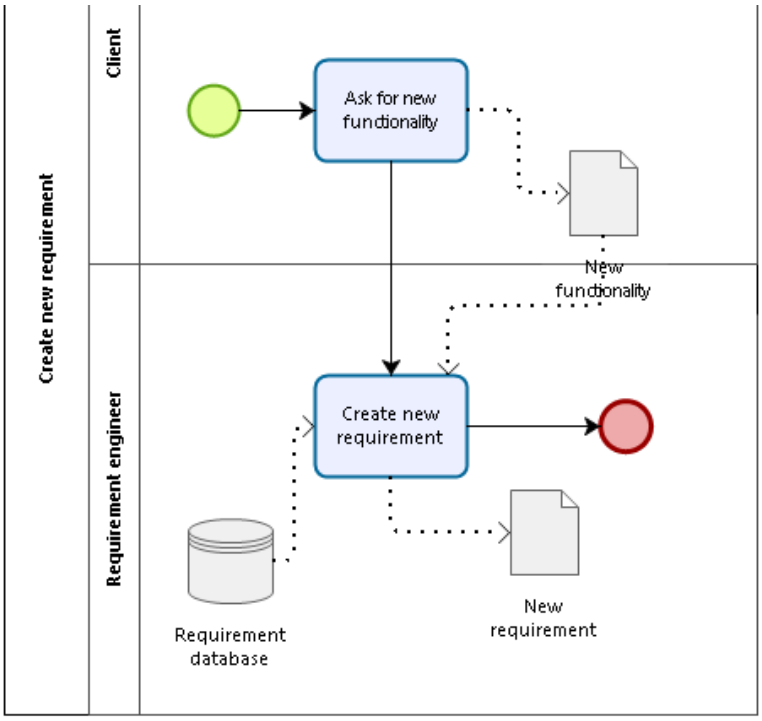

Figure 41 Create a new requirement

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal is necessary two functions requirements that are represented as Function (F) in Desiree:

G:Requirement be created

F:Create <object:new_requirement ><subject:requirement_engineer><parameter:new_functionality><where: requirement_database>

F: Ask <object:new_functionality><subject:cliente><target:requirement_engineer>

b.
Monitor agent

The monitor agent aims to achieve the goal of changes in requirements to be identified, and new requirements are identified. For this, he needs to monitor the requirements, when a change in requirement is made, the Requirement Management system is updated and sends an alert to the monitor agent, that needs to act and update the traceability matrix (Figure 42).

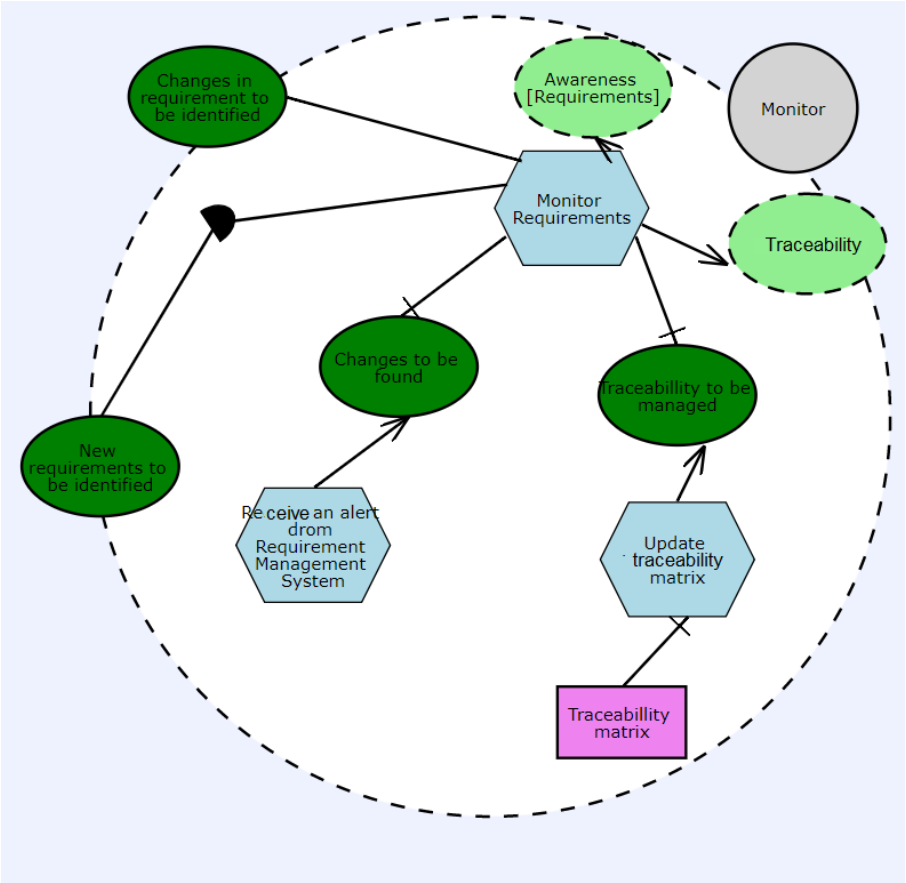

Figure 42 Monitor agent

- **Receive an alert**

When a change in requirement is identified an alert is sent (to a cell phone or email) to requirement engineer, then the requirement engineer needs to receive the alert and enter in the requirement database to download the changes. The process is shown in the BPMN diagram (Figure 43):

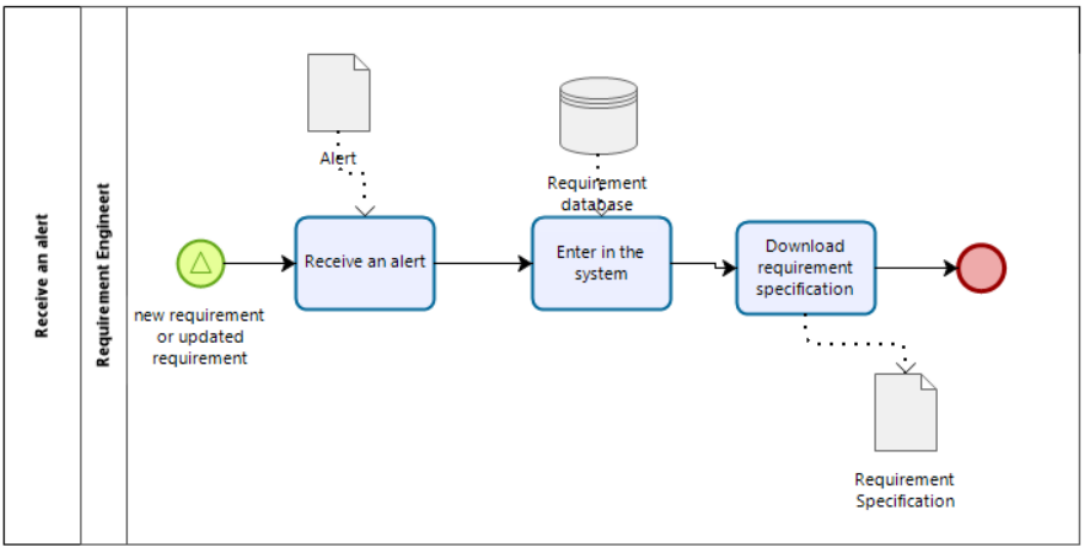

Figure 43 Receive an alert

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal there are necessary three functions requirements that are represented as Function (F) in Desiree:

G: Changes to be alerted

F:Receive <object:alert><subject:requirement_engineer>

F: Enter <object: requirement_database><subject:requirement_engineer>

F:Download<object:requirement_specification><subject: requirement_engineer >

- **Update configuration Matrix**

The maintainance of the traceability is necessary to update the configuration matrix, this task occurs in the same way as in the law changes (Figure 46).

c.
Analyzer agent

The analyzer agent aims to achieve the goal of the system always to be compliant. For this, he needs to make the data interpretation, to verify if the changes impact the software compliance, and for this he compares the updated law with the old version and analyzes the impacts of changes in SRS (Figure 46).

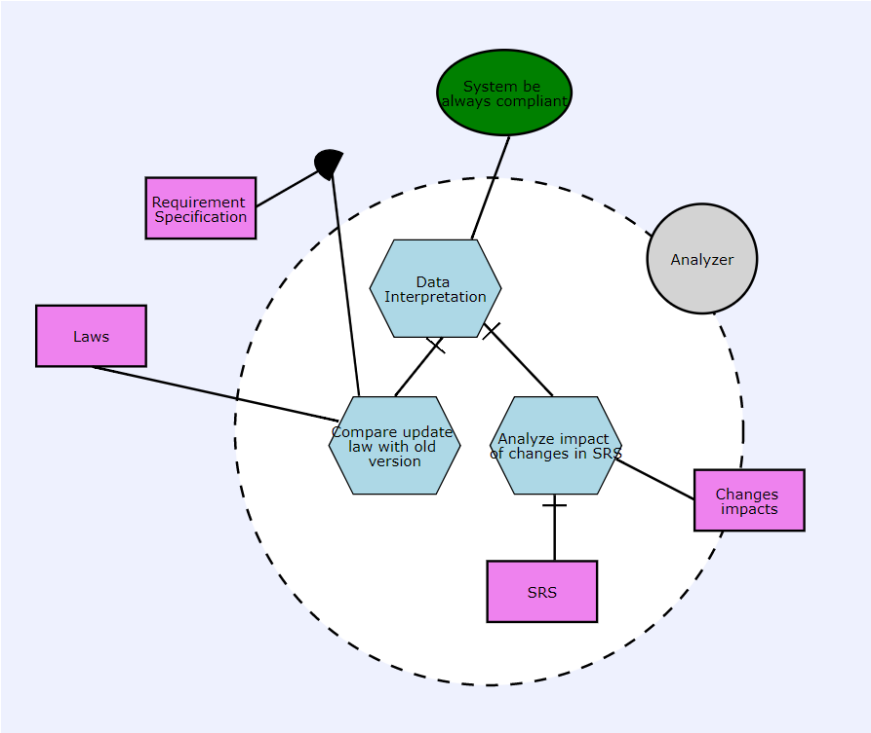

Figure 44 Analyzer agent

- **Compare updated requirement with the old version**

To understand the changes´ impacts first is necessary to read the new version and compare to the old one to verify where the changes are. Figure 45shows the BPMN diagram.

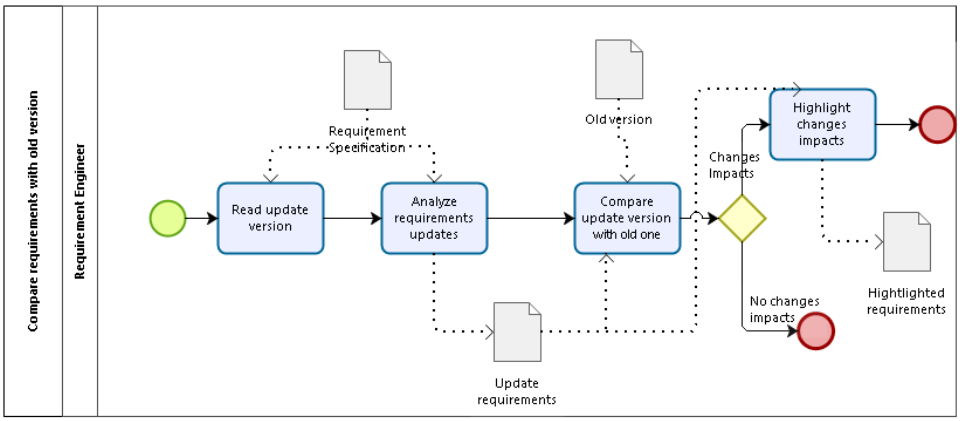

Figure 45Compare update version with the old version

In Desiree Language the task will be decomposed in two goals (G), to achieve these goals, there are necessary four function requirements that are represented by Function (F) in Desiree:

G: Requirements be compared

G: New requirement to be identified

F: Read <object:requirement_specification><subject:requirement_engineer>

F: Analyze <object: requirement_specification><subject:requirement_engineer>

F: Compare <object:old_version,update-requirement ><subject:requirement_engineer>

F: Highlight <object:Highlighted_requirement><subject:requirement_engineer><paramettter:update _version>

- **Analyze changes´ impacts in SRS**

When a change in requirement is identified it is necessary to see the impacts of the compliant software specifications. If it is a new requirement, it is necessary to compare them with all regulatory requirements identified in the legislation. If it is an update, it is necessary to verify if this update requirement has a regulatory requirement related to him and if the changes´ impacts the compliance. Figure 46 shows the BPMN diagram.

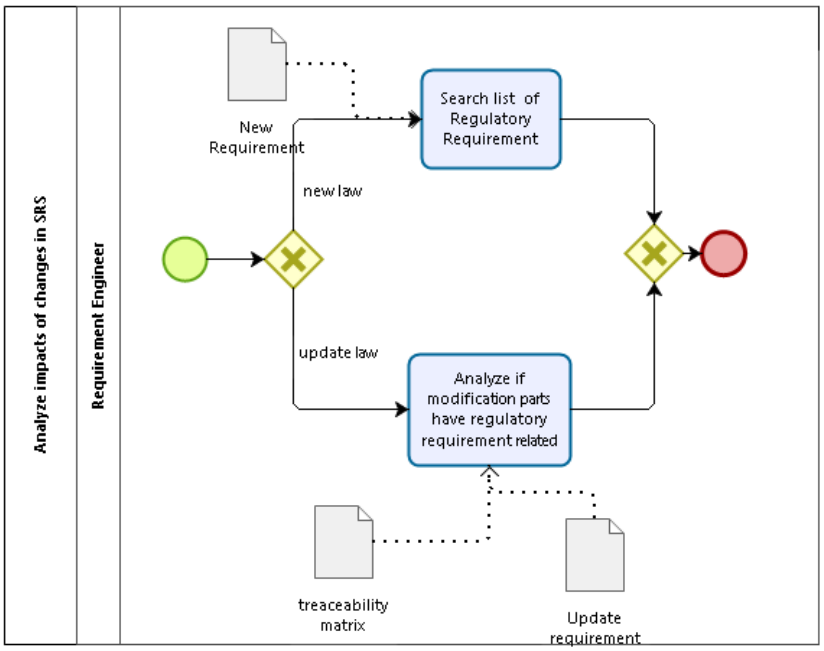

Figure 46 Analyze impacts of changes in SRS

In Desiree Language the task will be decomposed in two goals (G), to achieve these goals, we need 2 functions requirements that are represented by Function (F) in Desiree: G:Changes in requirement to be identified

G: Impacts be analyzed

F: Search <object:list_of_regulatory_requirement><subject: Requirement_engineer ><parameter:new_requirement>

F: Analyze

<object:modifications><Subject:Requirement_engineer><where:treaceabilty_matrix><

parameter:update_requirement>

d.
Planning agent

The planning agent will act the same as the planning agent acts in the case of Changes in legislation (Figure 32). The difference is that he will plan the compliance process execution focus on the requirement changes, not in law changes.

- **Plan Compliance Process Execution**

It is necessary to plan how the compliance process will be executed. In all list of regulatory requirements, for a specific regulatory requirement, or maybe will be not necessary to run the compliance process if the changes do not affect the requirements. Figure 47 shows the BPMN diagram.

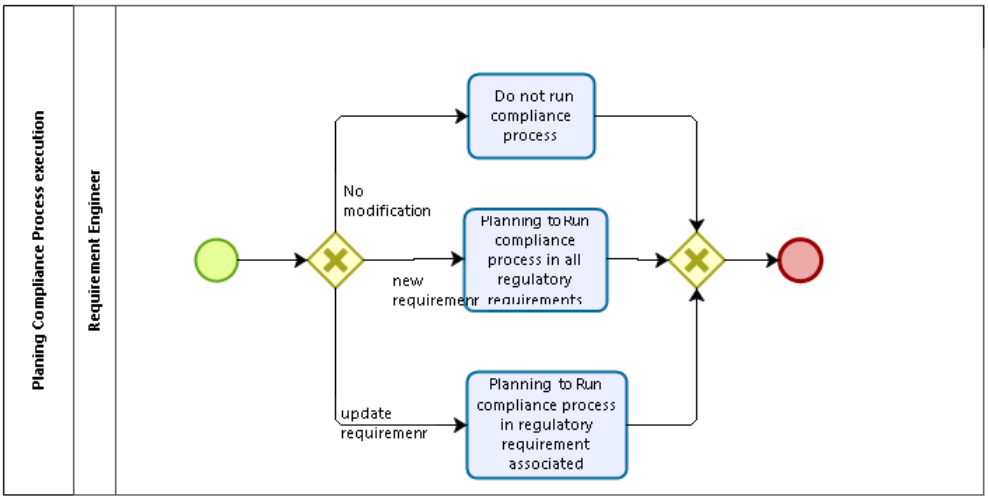

Figure 47 Planning Compliance Process Execution

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need three functions requirements that are represented by Function (F) in Desiree:

G:Compliant Process to be planned

F:Plan<object:complianceProcess><where:law><subject: Requirement_engineer >

F: Not run<object: complianceProcess><where:law><subject: Requirement_engineer >

F: Plan<object:complianceProcess><where:part_of_law ><subject:
Requirement_engineer >

e.
Executor agent

The executor agent aims to achieve the goal of Compliance to be Ensured. For this, he can execute the Compliance Process or an ad-hoc method to ensure compliance. And ad-hoc is low cost but has less precision and is time-consuming. In the case of executing the compliance process he doesn´t need to discover the requirements in legislation because the changes occur only in the requirements so, he will compare regulatory requirements with software requirements and resolve conflicts (Figure 48)

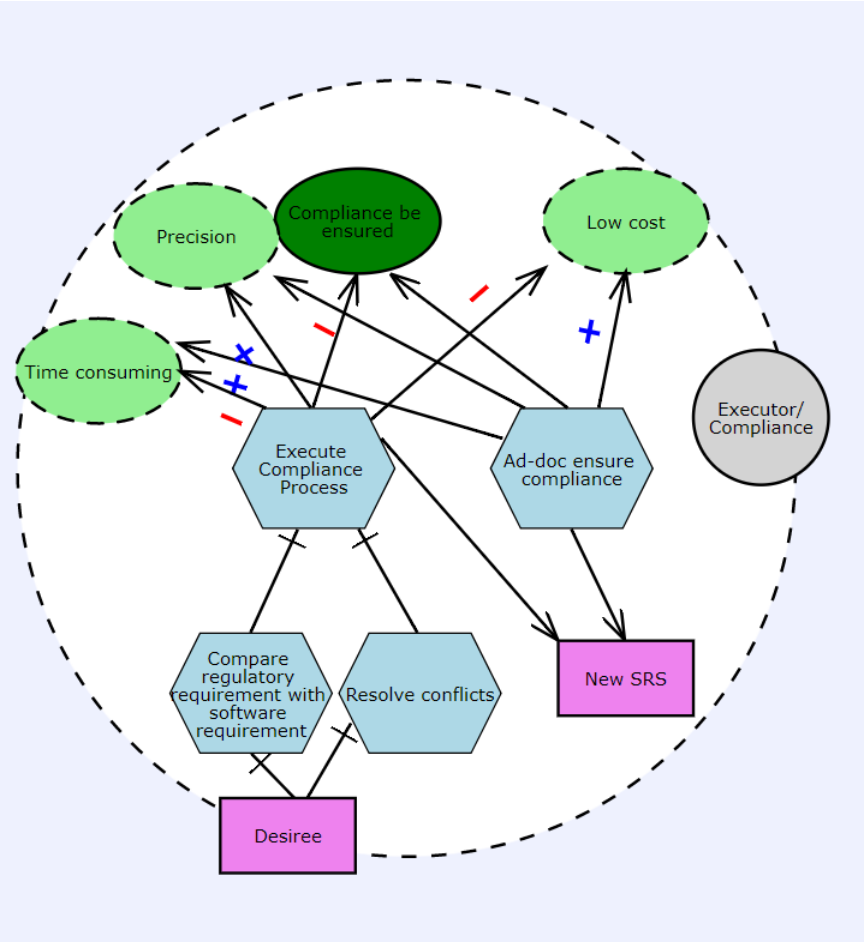

Figure 48 Executor agent

The tasks related to the Execute Compliance Process (Discover regulatory requirements, Compare regulatory requirement with software requirements, Resolve Conflicts) were already specified in 4.2 item.

4.6. Requirements and law changes

In the case of Law and Software changes, to achieve the goal of the system remains compliant, the monitor, analyzer, planning and executor agents will act external to software. When the changes are identified, the compliance process needs to be executed again (Execute Phase in MAPE cycle), and a new SRS will be generated. This new SRS needs to be implemented in software. The system representation for those cases is shown in Figure 49.

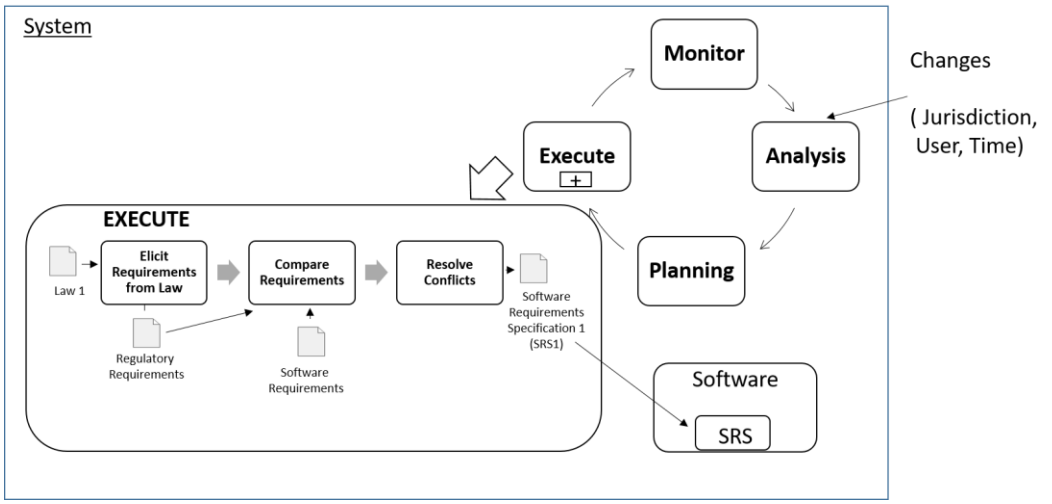

Figure 49 Compliance Monitor Process to change legislation and requirement

To show the sequence of steps of MAPE cycle, the SD Situation diagram (Oliveira, 2008) was used. The MonitorAgent is always active, monitoring the environment. When a change is realized, the Analyzer Agent will start, and when the Analyzeris finished, the Planning starts and at the end the Executor agent, that in those case is the execution of Compliance Process (Figure 50).

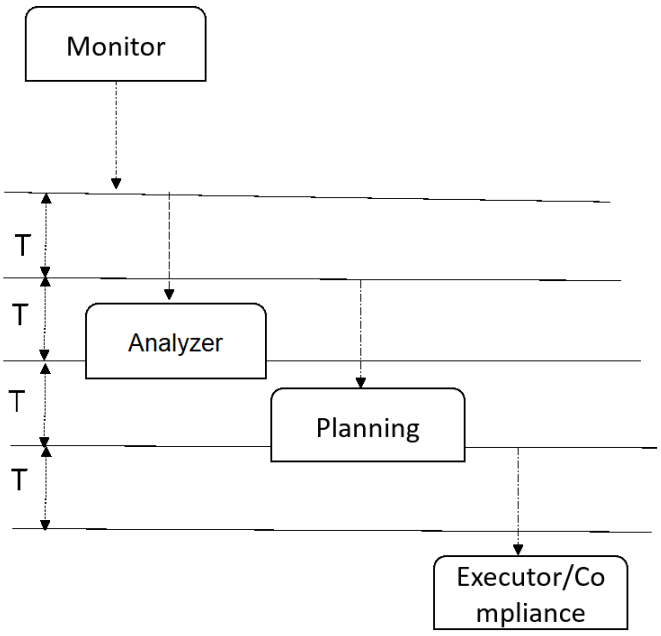

Figure 50 Situation Diagram

4.7. Compliance monitoring

With the advantages of mobile, the changes in the environment can be frequently and fast. Therefore, it is necessary to monitor the context change to the system adapt to them.

One key point is to identify what are the situations that the software needs to be aware. The system will need to adapt to all the situations. To help to understand all the situations that the system will need to adapt the context-aware treeis used.

a. Context awareness tree

Like the software, the law and the environment can change, because of this, it is necessary to define the context for our compliance process and the monitoring activities through which our system will remain context aware. We represent our Compliance Context using the Context Dimension Tree (CDT) (Orsi and Tanca 2011).

Context Dimension Tree (Orsi and Tanca 2011) are a graphical model for the context representation, and it is used to represent all possible context that you may have in an application. The root´s children are the context dimensions that describe the different characteristics of the context represented in black nodes; The white node describes the possible values that dimension can assume.

Our compliance process aims to represent the possible context situation which may influence the legislation that the software must be compliant with. In particular, the information that constitutes our compliance contexts (Figure 51) includes:

- **Geographic jurisdiction:** The location has a significant influence on legislation. Each country and in some place each state can have different legislations that needs to follow. If the user or the operating software system change the location, probably the legislation will change too and consequently the regulatory requirements.
    - **The geographic jurisdiction of the user**, such as being located in “Italy,” or in “Trento, Italy”;
    - **The geographic jurisdiction of the operational software system**; for example, the system may be operating in “Germany”;
- **Time of user accessing the system**: We can have different rules for different periods, like rules to follow on weekdays or weekends or regulations to be followed during the day and rule for the night. In this case, probably is not all the legislation that will change, but perhaps some part of the law needs to be considered or not. For instance, the user driving a vehicle overnight;
- **The user**, depending on the type of user, different legislation need to be followed. For public organizations, the laws that need to be applied can be different from the private ones. For an underageperson, the rules can be different.

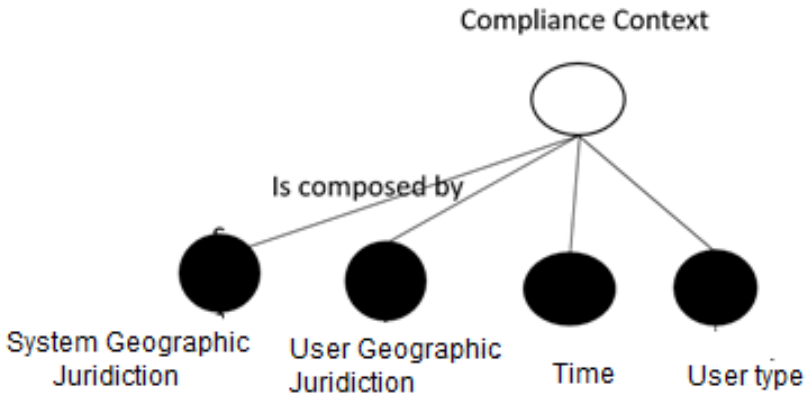

Figure 51 Compliance context tree

The system needs to remains aware of changes for each type of information included in the compliance context:

- Jurisdiction – The software needs to have a way to verify where the system is and where the user accessing the system is. To test the location is necessary to use systems of geographic position such as GPS, LPS.
- Time – To confirm the time that system is being accessed in we can use the networks that are available like our mobile does.
- User type – to discover the user type it is necessary to have a user control access, wherein the user´s profile have the information of user type.

b.
Context monitoring

The Context monitoring is continuous. Therefore, we will use the feedback loop to be aware of the context change to promote that the software remains compliance. The software should always be aware of possible changes, so it will always be monitoring the physical environment, for instance, using sensors, such as GPS. If the monitored values show a change, the software needs to adapt.

In this case, the system is composedof the software, the monitor process, and the compliance process (Figure 52). The changes are external to the system. Also, the compliance process is external to software and needs to be executed at requirement time. Before the software is developed it is necessary to analyze all the possible situations that the software needs to adapt. For each situation one SRS is generated, and when the situation is identified, this SRS will be activated at runtime.

The SRS will be an input for the MAPE planning phase. The system will always be monitoring the environment, when a change is identified (analysis phase) it is necessary to verify if an adaptation is necessary (which SRS need to be activated). The executor will activate the available SRS for the new situation, and its compliant code will be executed.

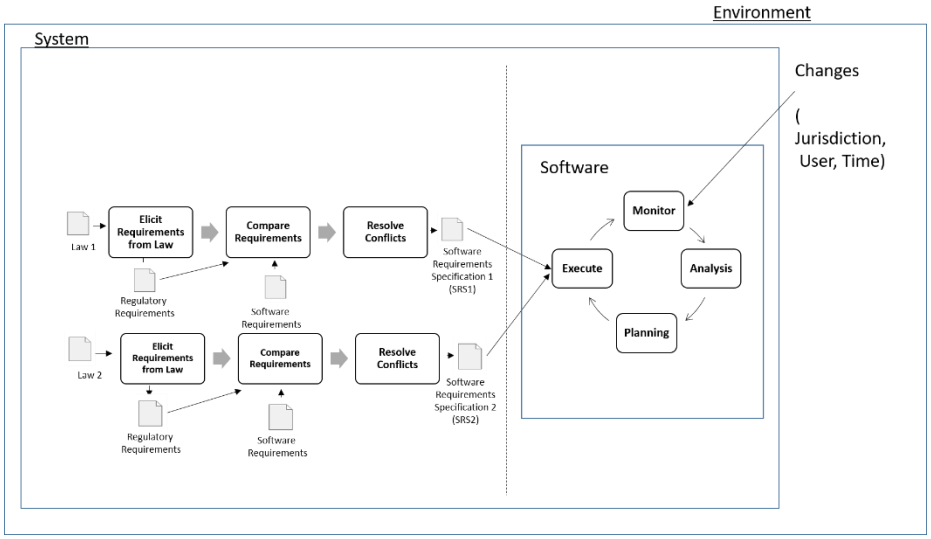

Figure 52Compliance monitor process for environment change

In this case, we have the Software Requirement (SR) and a different set of laws to be applied, one or more for each situation that the software needs to adapt to, so we will have an OR situation between laws form different jurisdictions. So we can say to encourage Compliance is a combination of software requirement (SR) and a set {} of laws ofthe differentjurisdictions (Ljur) represented as C = SR AND {LjurA} OR {LjurB}.

Such situation introduces a variability on the software, and it brings to bear the analogy of software product line, in which a product may have different variants. In this case, each one possible SRS may be put together as product line requirement specification (PRS)(Faulk, 2011) (Insfran et al., 2014)(Yu et al. 208).

A product line (Figure 53) is a family of products which have featured in common. PRS explicitly represents the family´s standard requirements as well as the Software Requirement Specification (SRS) will represent the variation in family members. Therefore, we use this idea to represent the Software Requirement Specification (SRS) needed to guarantee compliance in the case of context change.

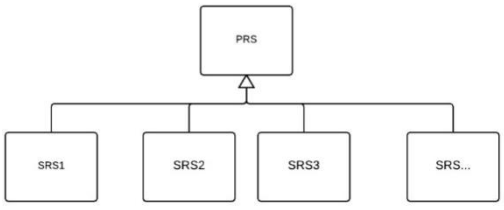

Figure 53 Product line requirement specification diagram

The system needs to adapt to legal requirements to accommodate legal variability. To have a compliant set of requirements for each context change, it is necessary, for each related legislation, that the actor Compliance acts (Figure 54). As such, if there is more than one possible context, the possible SRSs resulted from the compliance process will be organized as a product line requirement specification. The compliance Process will run and generate all the possible SRS. When the system is running, the monitoring phase

begins. If a change is identified, the Analyzer, Planning and Executor will act to adapt the software for the changes.

In this case, we have three actors (Person, Environmentand Software) and the five agents (Monitor, Analyzer, Planning, Executor and Compliance agent). In this case, the four agents are part of the software, since the MAPE occurs at runtime. The compliance agent is external to the software, the requirement engineer (person) that will perform the compliance process. The environment is responsible for the chances. The software and person are part of the environment (Figure 54).

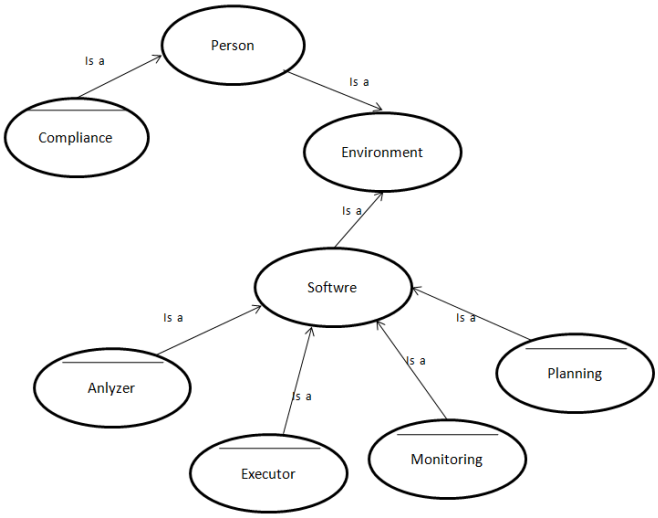

Figure 54 SA diagram for context change

Software that needs to autoadapt has awareness as a fundamental requirement. De Souza, 2014 proposes to use goals models, to help the software to perceive the environments and its changes. This work reuses the SRConstruct of consciousness (de Souza, 2014) since it is necessary for monitoring the environment and adapting it to the changes (Figure 55).

To promote that the software is always compliant, it is necessary to monitor some signals in the environment (location, time and user) and (re)act accordingly. Therefore, the corresponding behavior in response to the underlying context situation can be defined in a deterministic way: based on values of monitored variables it is possible to decide the best action to be taken in all cases (what SRS will be used).

The method for operationalization of the context awareness can be split into two steps: (i) data acquisition (equivalent to the MAPE monitoring function), and (ii) data interpretation (equivalent to the MAPE analysis and planning function). The acquisition is the function that returns data values that represented the situation in a determinate instance of time.

The monitor agent will always be running. If the monitor agent realizes some changes in the environment, the analyzeragent needs to interpret and to confirm if it is a situation where the context changes. If it confirms the changes,the planning agent will plan the actions for the system adapts to the new situation. The Execute agent will activate the correct SRS to the new situation ensuring the compliance.

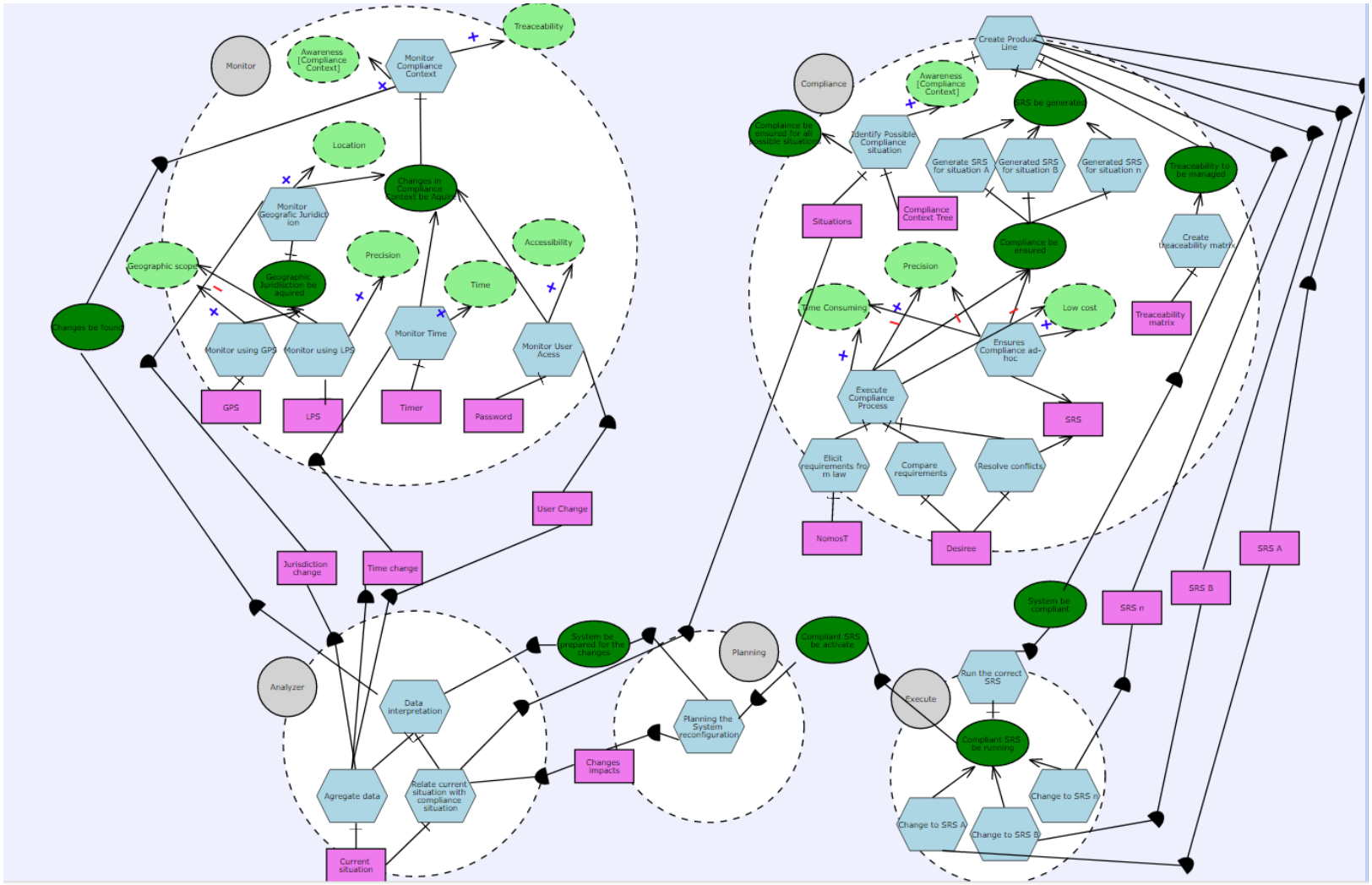

Figure 55 SR Diagram for changes in context

For each task (which is a leaf in the diagram) we define how it will work using BPMN diagram and the Desiree Software Specification.

a.
Monitor agent

The monitor aims to achieve the goal of Changes to be found. For this, he needs to monitor compliance Context. For this is necessary to monitor the geographic jurisdiction that can be done using GPS or LPS, Monitoring time and Monitoring the User access. (Figure 56)

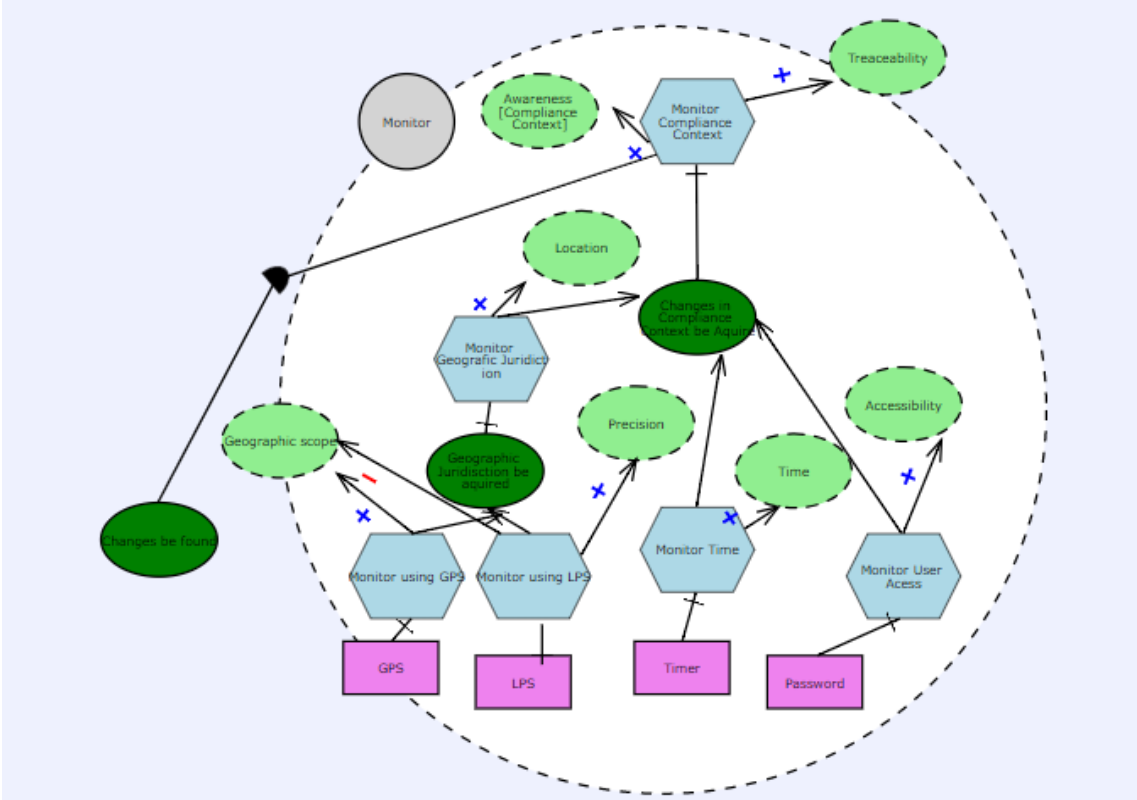

Figure 56 Monitor agent in context change situations

- **Monitoring using GPS**

The software needs to send the GPS signal to the satellite, the satellite will return the satellite information, and the software can calculate the current position. This current position will be compared with the old one and discover if there´s a change in location. The process is showed in the BPMN diagram (Figure 57)

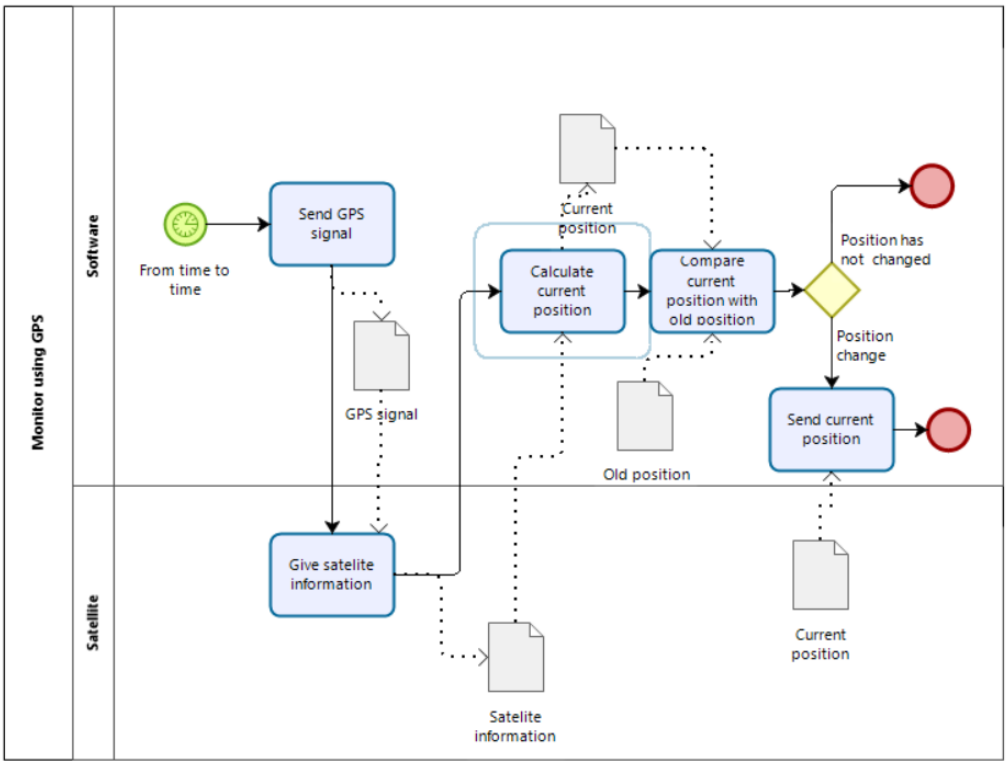

Figure 57 Monitoring using GPS

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need five functions requirements that are represented as Function (F) in Desiree:

G: Position to be acquired

F: Send <subject:software ><target: satellite_system><object:GPS-signal><when:time_to_time>

F: Give<subject:satelite_system><target: software ><object:satellite_information>

F: Calculate <subject: software ><object:current_position><parameter:satellite_information>

F: Compare <subject: software >< object:{current_position, old_position>

F: Send <subject:software><object:current_position>

- **Monitoring using LPS**

LPS position system uses the near equipment to calculate the current position. The system picks the position of the near equipment and does a calculation to determinate its position.

So, the software needs to search for near equipment. The equipment will send the position data. The software can calculate the current position. This current position will be compared with the old one and discover if the location have changed. Figure 58 shows the BPMN diagram.

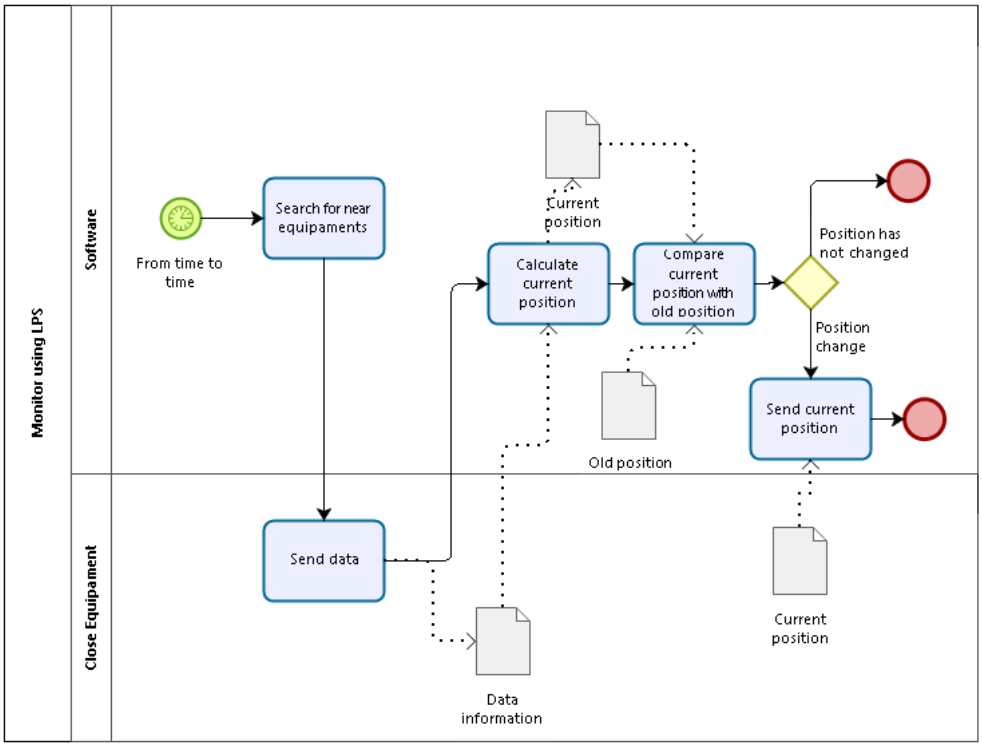

Figure 58 Monitoring using LPS

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need five functions requirements that are represented as Function (F) in Desiree:

G: LPS Position to be acquired

F: Search<subject:software><object:near_equipament><when:time_to_time>

F: Send <subject:close _equipament ><target:software><object:data_information>

F: Calculate <object:current_position>< parameter:data_information><subject:software>

F: Compare <subject: software >< object:{current_position, old_position>

F: Send <subject:software><object:current_position>

- **Monitoring user**

From time to time, the software needs to verify who is using the system and search for their profile. This current profile will be compared with the old one and discover if the user has changed. Figure 59 shows the BPMN diagram.

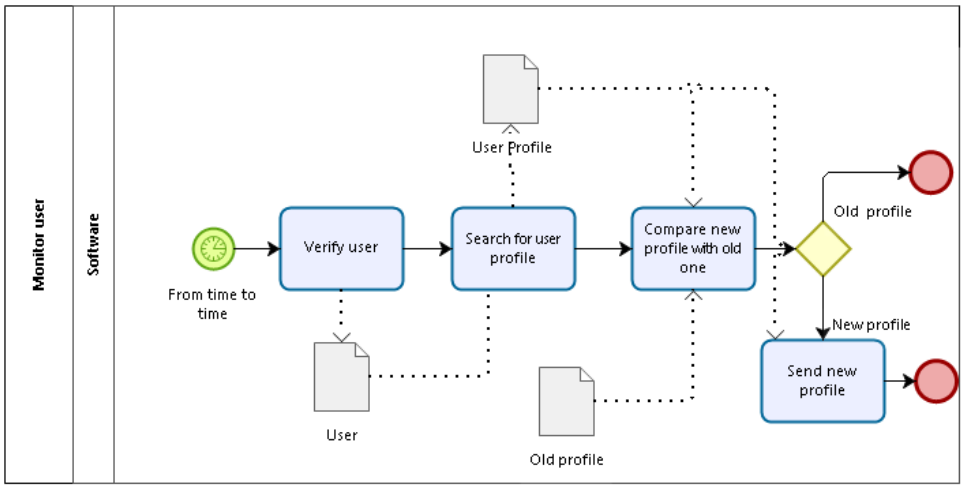

Figure 59 Monitor user

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need two functions requirements that are represented by Function (F) in Desiree:

G: Change user to be discovered

F: Verify <subject:software><object:user>

F: Search <subject:software><object: user_profile><parameter:user>

F: Compare <subject:software>< object:{user_profile,old_profile}>

F: Send <subject:software><object: user_profile>

- **Monitoring time**

From time to time, the software needs to verify the system date; the current data will be compared with the old one, to verify if it has changed. Figure 60 shows the BPMN diagram

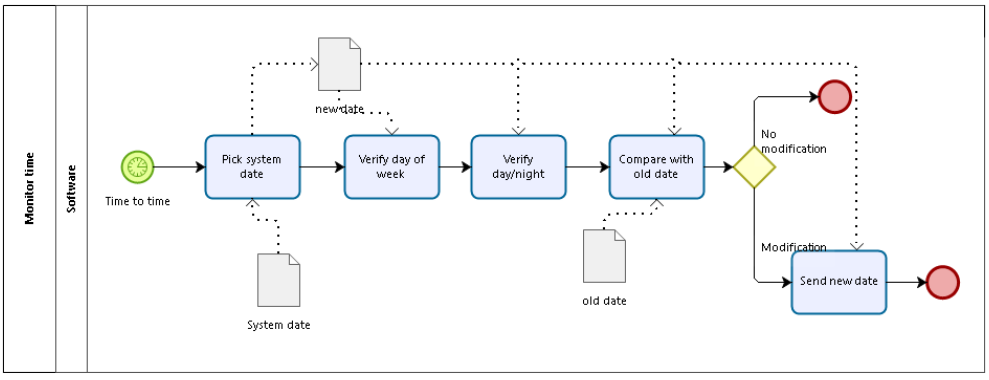

Figure 60 Monitoring time

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal there are five functions requirements that are represented as Function (F) in Desiree:

G: Change Time to be discovered

F: Pick <subject:software><object: system_data><when:time_to_time>

F: Verify<subject:software><object: new-data>

F: Verify<subject:software><target: new-data><object: day_of_week>

F: Verify<subject:software><target: new-data><object: day/night>

F: Compare <subject:software>< object:{new_data, old_data>

F: Send<subject:software>< object:new_data >

b.
Monitor agent

The monitor aims to achieve the goal of the system to be prepared for changes. For this, he needs to make the data interpretation, that consists of aggregating data (verifying the changes in jurisdiction, user and time) and relate the current situation with compliance situation (Figure 61).

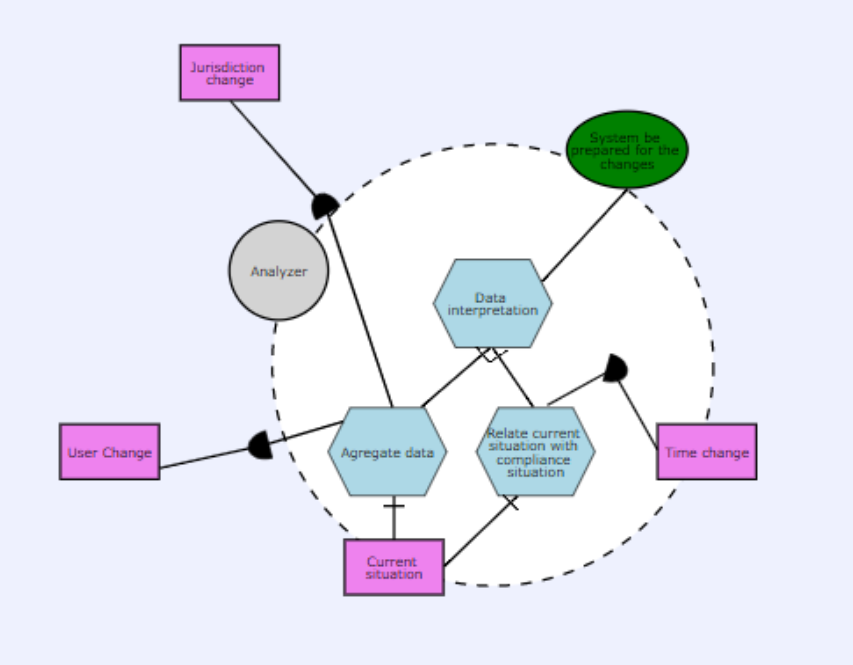

Figure 61 Analyzer agent in context change situation

- **Aggregate data**

All the changes will be received and need to be combined to generate the new situation that the software may need to adapt. Figure 62 shows the BPMN diagram

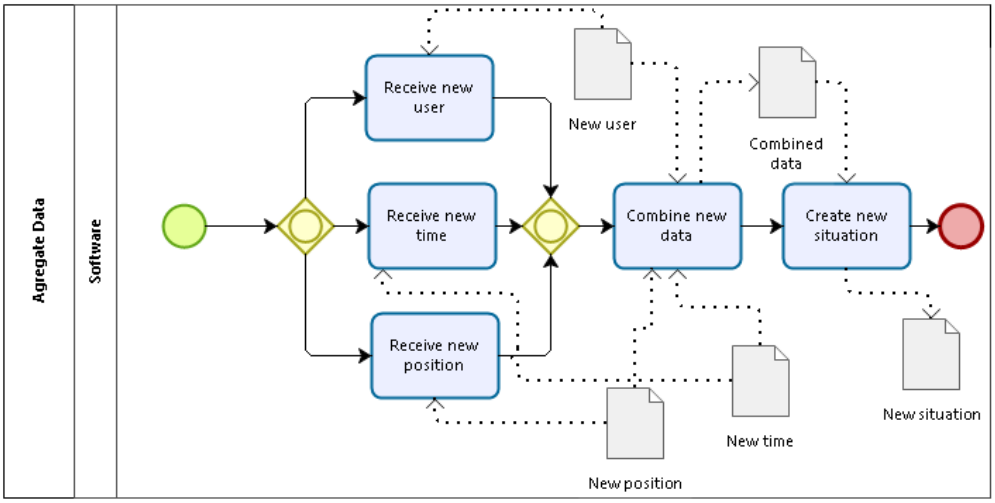

Figure 62 Aggregate data

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need five functions requirements that are represented by Function (F) in Desiree:

G: New situation to be created

F: Receive <object:new_user><subject:software>

F: Receive <object:new_time><subject:software>

F: Receive <object:new_position><subject:software>

F:Combine <object:combined_data>><parameters:new_data,new-user;Current_position>t.

F: Create <object:new_situation><parameters:Combined_data><subject:software>

- **Relate the current situation**

Situations are the different combinations of user, time and location that the environment can achieve and the software will need to adapt.

After the new situation is discovered it is necessary to find them in the list to see what SRS correspond to it. Figure 63 shows the BPMN diagram.

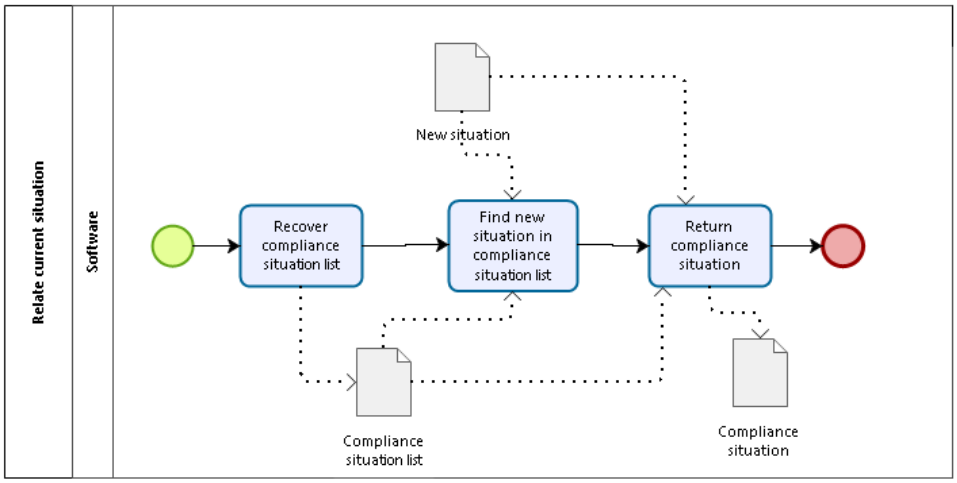

Figure 63 Relate a current situation

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need three function requirements that are represented as Function (F) in Desiree:

G: Situations to be related

F: Recover <compliance_situation_list><subject:software>

F: Find <object:compliance_situation ><parameters:new_situation; compliance_situation_list><subject:software>

F: Return<object:compliance_situation ><subject:software>

c.
Planning agent

The planning agent aims to achieve the goal of compliant SRS to be activated. For this, he needs to plan the system reconfiguration for the new SRS (Figure 64).

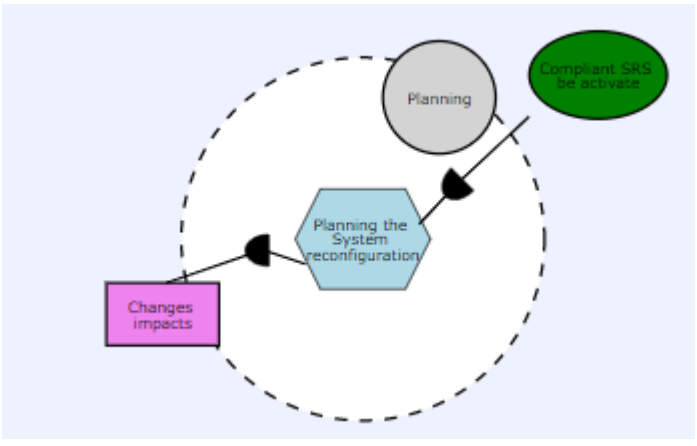

Figure 64 planning agent for the context change situation

- **Planning system reconfiguration**

For planning the software reconfiguration, it is necessary to find in the matrix the correct SRS that is related to the new situation. Figure shows the BPMN diagram

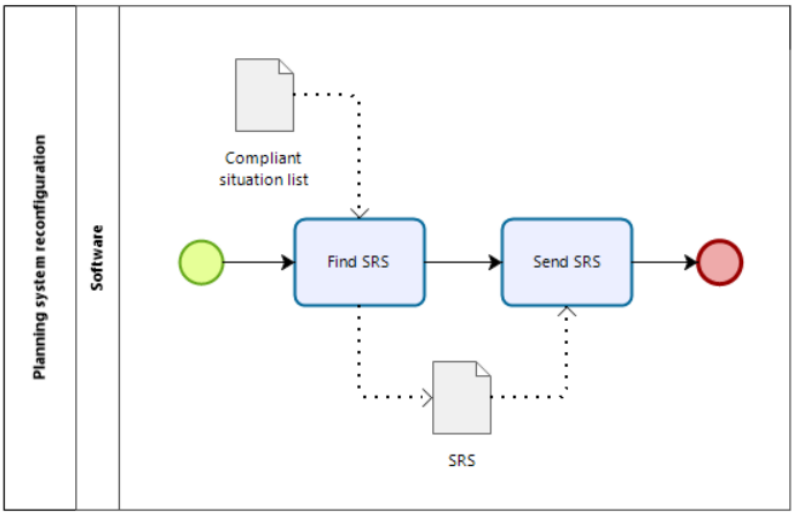

Figure 65 Planning system reconfiguration

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need two functions requirements that are represented as Function (F) in Desiree:

G: Compliant SRS be found

F: Find <object:SRS ><parameters:compliance_situation; list_SRS ><subject:software>

F: Send <object:SRS ><subject:software>

d.
Executor agent

The planning agent aims to achieve the goal of compliant SRS to be compliant. For this, he needs to run the correct SRS (Figure 66).

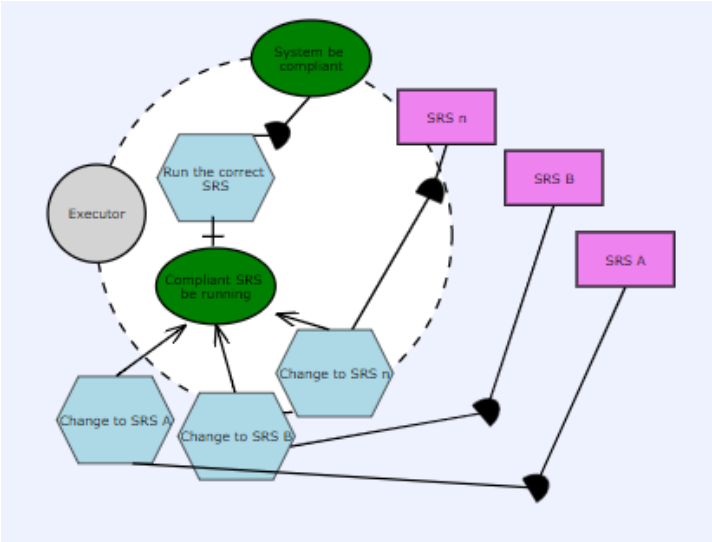

Figure 66 Executor agent for the context change situation

The tasks related to the Execute Compliance Process (Discover regulatory requirements, Compare regulatory requirements with software requirements, Resolve Conflicts) were already specified in 4.2 item.

- **Change to SRS**

The system needs to change to the correct SRS to be compliant with the new situation. The system will receive the new situation, and the SRS will reconfigure the system. Figure 67 shows the BPMN diagram.

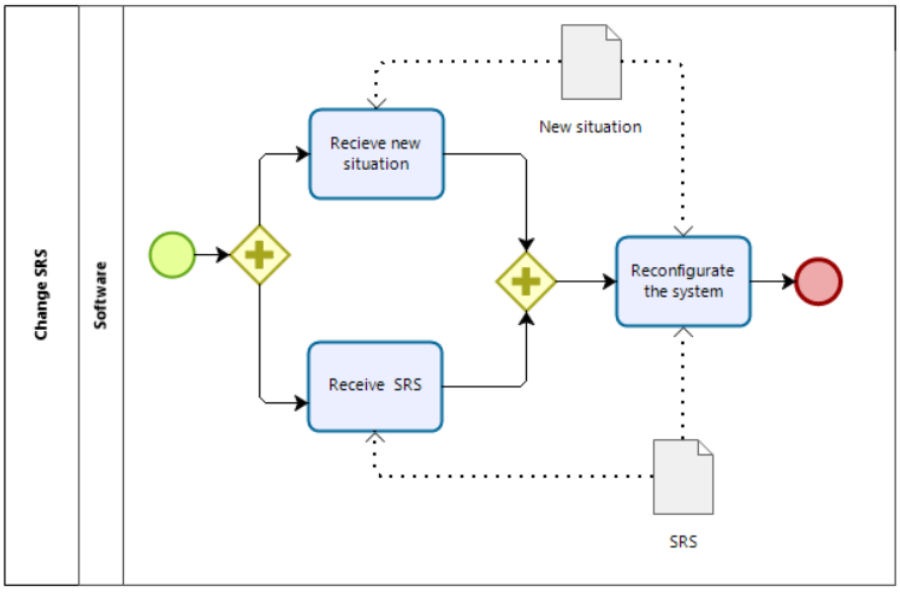

Figure 67 Change SRS

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need three functions requirements that are represented by Function (F) in Desiree:

G: SRS to be configured

F: Receive <object:new situation><subject:system >

F: Receive <object:SRS ><subject:system >

F: Reconfigure<object:system><subject:system ><parameters:SRS>

e.
Compliance agent

The compliance agent aims to achieve the goal of Compliance to be Ensured. In this case, before running, the compliance process is necessary to identify all possible compliance situations and to generate the SRS for each situation running the compliance process. Also it is necessary to create the traceability matrix, to know which SRS is related to each situation (Figure 68)

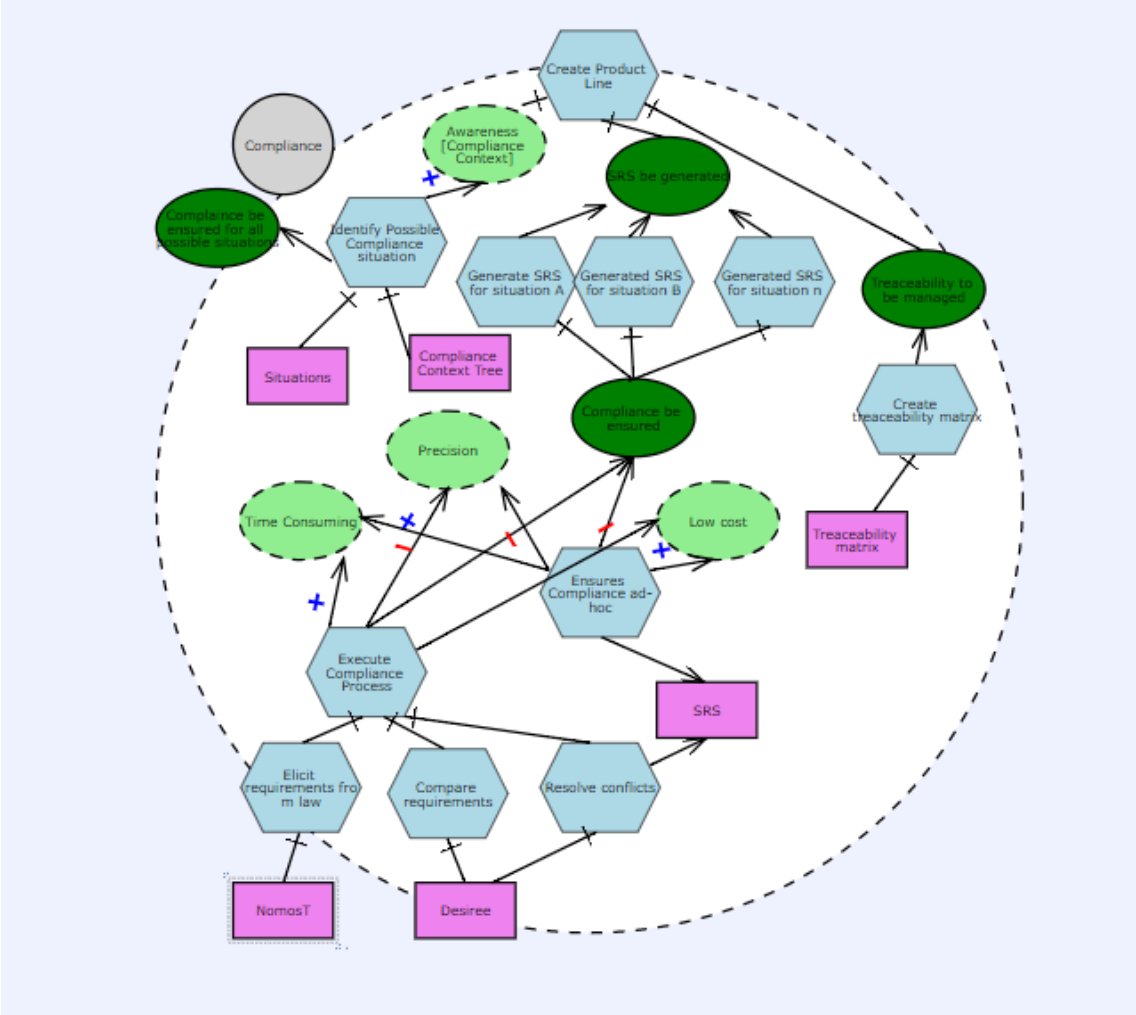

Figure 68 Compliance agent for context change situation

- **Identify possible compliance situations**

It is necessary to analyze and identify all the possible situations that the software can meet. The situations will be a combination of the variations of locations, time and users. Figure 69 shows the BPMN diagram.

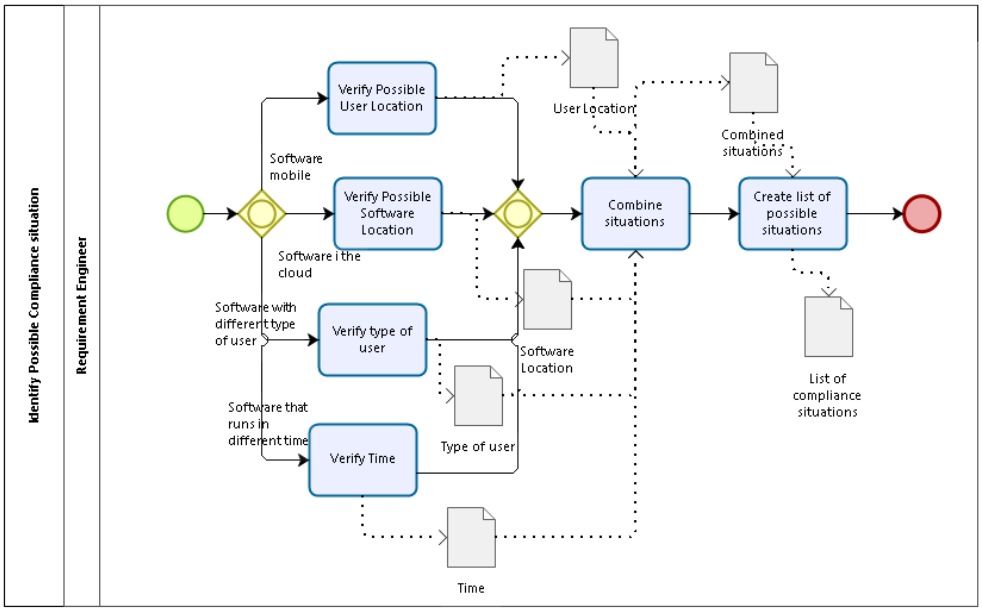

Figure 69 Identify possible compliance situation

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need five functions requirements that are represented as Function (F) in Desiree:

G: Possible situations to be generated

F:Verify <object:possible_user_locations ><subject:requirement_engineer>

F:Verify <object:possible_software_locations ><subject:requirement_engineer>

F:Verify <object:Possible_user_type><subject:requirement_engineer>

F:Verify <object:possible_time><subject:requirement_engineer>

F: Combine <object:situations><parameters:{ PossibleUserlocations , PossibleSoftwarelocations, PossibleUserType , PossibleTime }><subject:requirement_engineer>

F:Create <object:list_possible_situations><subject:requirement_engineer> <parameters:combined_situations>

- **Create Configuration Matrix**

To be easier to trace the relation between situations and the related SRS, itis necessary to create a configuration matrix.Figure 70 shows the BPMN diagram.

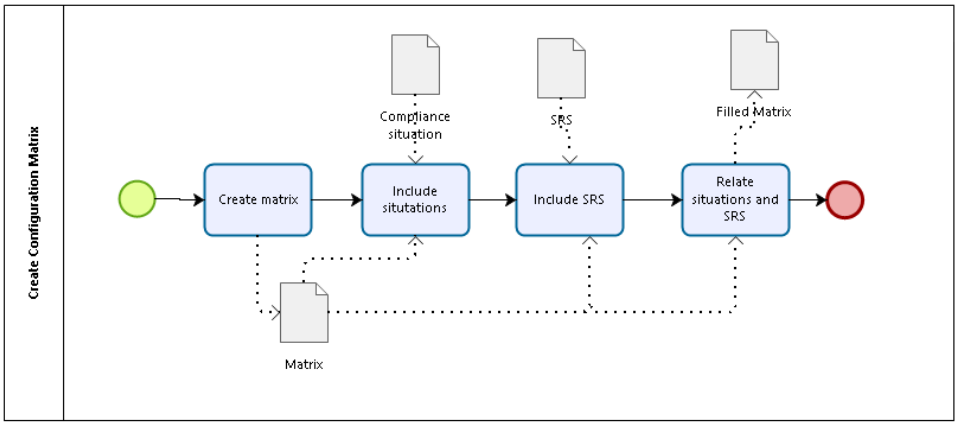

Figure 70 Create configuration matrix BPMN

In Desiree Language the task will be decomposed in a goal (G), to achieve this goal we need four functions requirements that are represented by Function (F) :

G: SRS and situations to be traced

F: Create <object:matrix><subject:requirement_engineer>

F:Include <object: situations><where:matrix><subject:requirement_engineer><parameter:compliance_situation>

F: Include <object: SRS><where:matrix><subject:requirement_engineer>

F: Relate <object:SRS_situations ><where:matrix><subject:requirement_engineer><parameters: matrix>

The tasks related to the Execute Compliance Process (Discover regulatory requirements, Compare regulatory requirements with software requirements, Resolve Conflicts) were already specified in 4.2 item.

The SDSituation model was used to show the tasks in a sequence of time. Differently from the previous situations, the Monitoring begins after the Compliance acts. First, it is necessary to analyze all the possible situations that can happen and run the compliance process for all legislation that can affect the software. After the system having all the SRS generated and organized as a product line and the software developed, the Monitor agent can starts to act. When a change is verified, the analyzer and plan agent will act, followed by the Configurator. The Analyzer starts to achieve its goals after an interval of the Monitor when it already has the information about the context changes. Figure 71 shows a SDSituation Model for the case of context change.

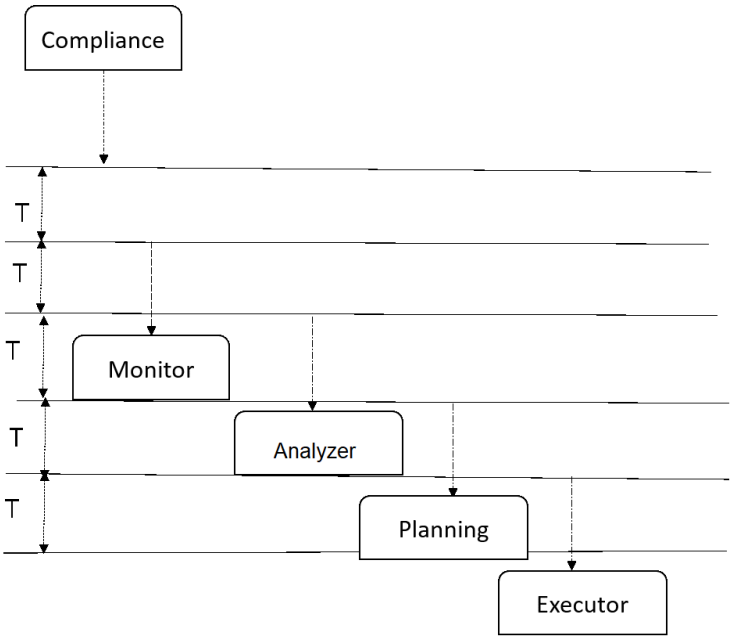

Figure 71 SD Situation for Context

4.8. Chapter summary

This chapter presents then Eunomia Framework, that is composed of two parts, the Compliance Process and the Compliance Monitor. First, we presented the conceptual model, explaining all the concepts used in the proposal, the relation among them, and how the two parts of the framework are related.

Then, the Compliance Process is explained. The Compliance process is a tool-supported process that helps to elicit requirements from the law, compare them with software requirements and resolve the possible conflicts. The Compliance Process used the NomosT tool to extract the requirements from the law and the Desiree Framework to compare and resolve the possible conflicts that may arise. The Compliance process is a systematic method to generate the compliant software requirement specification from a list of software requirement and a list of laws.

The second part is the Compliance Monitoring; compliance is not a static situation, the law, the requirements and the context of the software are in a constant change, so there are necessary mechanisms that promote the continuous compliance. The Compliance Monitoring is based on agents and product line to help to ensure the continuous compliance. The monitoring is represented using agents because the environment that the software is running is not predictable, the subjectto quick changes and the software need to decide and to adapt to the situations to achieve the goals. The agents are model in i* to show the variability and the rationale to achieve the goals. To represent the requirements of the behavior of each agent we use the BPMN and the Desiree Notation, in this way we have the specification to develop the agents. The product line was used to organize the different software requirement specification, for each different situation a different list of requirements needs to be implemented. Some requirements are present in all list of requirements, and others are specific for each situation.

5
Evaluation

This chapter presents the evaluations of the proposal, the results and lessons learned. First,an early evaluation of the Compliance Process using the Italian Privacy Law is presented. Then to evaluate the Compliance Monitor two examples are shown, one of location change and other about the context, chasing an automated car.

5.1.
An evaluation of compliance process using Italian law

To evaluate the Compliance Process, a case study has been conducted. The evaluation aims to compare the proposed process with an ad hoc one. This study was intended to answer the research questions:

RQ2: What can requirement engineers do to find regulatory requirements from laws in a systematic way?

RQ3:What can requirements engineers do to verify software requirements against regulatory requirements in a systematic way?

RQ4:What can requirements engineers do to resolve possible conflicts between regulatory requirements and software requirements in a systematic way?

To answer the questions, we evaluate the effectiveness of the tool-supported process in reducing human effort, and on the other hand, we evaluate the quality of the output. We consider that better quality output entails greater compliance.Two measures were used. The first measures the effort (time) it takes to elicit the regulatory requirements and compare them with the pre-existing ones (the effort measure). The second measure evaluates the correctness of the new set of requirements generated relative to a gold standard and the number of requirements found (number of regulatory found, number of conflicting found, number of conflicting solved). These measures are applied to both: the compliance process proposed and an ad-hoc process.

Context: The context of the case study describes the conditions surrounding its execution (Travassos, Gurov, Amaral, 2012):

(a) Materials: The case study was conducted using a data set developed by DePaul University Master's students as part of a 10-week RE course project, which conducted a realistic Nursing Scheduler case study. The list of software requirements was selected from this set, along with Chapter 3 (English version) of the Italian privacy law, a list of bind terms and a list of keywords. The list of requirements and Chapter 3 are available on Annex 1 (http://bit.ly/2abxqrr)

(b) Participants: the five participants (Table 7) were University of Trento's Ph.D. Computer Science students and experts in requirement engineering. In this study we did

not have the presence of a law consultant; instead, we mitigated the problem by giving participants the list of keywords and the list of the binding terms.

(c) Tasks: Participants had to find requirements in the law and compare them with the pre-existing requirements, checking for conflicts. If conflicts are found, the participants needed to resolve them. The author of this thesis was responsible for building the gold standard;

(d) Groups: Participants were allocated into two separate groups in alphabetical order: Group A was composed of participant 1 and 2 and Group B by participants 3, 4 and 5. Group A was asked to generate the new set of requirements without using the tool-supported process (ad-hoc process). Group B was asked to generate the new set of requirements using the process. All the participants (Group A and Group B) did the task individually. Group B had 30 minutes of training about the method and the tools (NomosT and Desiree). Group A did the task in an ad-hoc way without any training, using their background in requirements engineering and common sense.

It is important to note that all the participants knew about goal modeling (a prerequisite for using the Desiree Tool). The first author of this thesis supervised all participants while they conducted their respective case study.

Table 7 Participants

Participant	**Requirement Experience (Academia)**	**Requirement Experience (Business)**
Participant 1	5 years	4 years
Participant 2	10 years	10 years
Participant 3	5 years	5 years
Participant 4	9 years	7 years
Participant 5	6 years	5 years

To answer the metric about the effort, we evaluated how much the proposed process improves performance concerning the ad-hoc process by measuring the time taken to elicit, compare requirements and resolve conflicts for both processes. To answer the metric about quality we compare the quality of the results of both processes when compared with the gold standard, produced by the first author. The gold standard was used pairwise with the sets of resulting requirements produced by both strategies (proposed process requirements and ad-hoc process requirements).

Table 8 condenses the results by showing the time spent and how many requirements and conflicts were found in comparison with the gold standard. We show this as a ratio (participant result/expected result) for each participant. Table 8- shows for each participant, the group, if the proposed process was used, the time to finish for each participant, the number of functional requirements found, the number of non-functional requirements found, and the number of conflicts found.

Table 8 Results by participants

Participant	Proposed Process	Time (min)	New FR	New NFR	Conflicting Requirement
Participant 1 (Group A)	No	68	4/10	0/4	2/2
Participant 2 (Group A)	No	61	7/10	0/4	1/2
Participant 3 (Group B)	Yes	73	7/10	4/4	2/2
Participant4 (Group B)	Yes	84	9/10	3/4	2/2
Participant 5 (Group B)	Yes	69	10/10	4/4	2/2

The study has used seven pages of law. In a larger document, the time result may be different, because of the search effort. It is important to remember that the time includes the overhead of translating texts into NomosT and Desiree. Even with these difficulties, the time of Group A is very similar to the time of Group B.

Concerning discovering new requirements, Table 9- shows an improvement in the case of using the Compliance Process. Although to find conflicting requirements, there is not a big difference using or not using the method. Observing the new requirements generated by the ad-hoc process, the participants could not find any non-functional requirement; this could happen because people are not so concerned with non-functional requirements. Participants using the method, on the other hand, had to pay attention to non-functional requirements, not functional ones.

The participants who used the method discovered almost twice as many requirements, explaining the extra time needed. In the conflict resolution, Group A dropped requirements and added new ones, but they did not modify existing requirements. This may have occurred because the idea of strengthening or weakening requirements is not so widespread.

Another observation is that the participants are not using the proposed method, checked compliance using the text of the law. They compared the software requirement with a section of the law, they did not try to extract requirements and then compare it. This may be an explanation for the lower number of requirements found by Group A.

Table 4 illustrates the conflict resolution performed by participant #5, using the tool supported method.

Table 9 Example of conflict resolution

Regulatory Requirements (RS)
1. unauthorizedAcessPrevention(studentRecord) = safe
Software Requirements (SR)
1. displaySummary<object: studentRecord> 2. display <object: studentNameAndGrade> 3. display <object: studentFrequency>
Conflict resolution (Strength the SR)

1. displaySummary<object: studentRecord><target: ONLY authorizedUser> 2.display <object: studentNameAndGrade>< target: ONLY authorizedUser> 3. display <object: studentFrequency>< target: ONLY authorizedUser>

Student Records (that have information about grades and frequency) constitute private information. The law says that private information should be accessed only by an authorized user (regulatory requirement 1). The software requirements (1, 2 and 3) say that instructors need to display student record, student name, grade, and frequency, so we have a conflict. To resolve it, participant 5 adds a constraint, saying that the student's records, name, grades and frequency can be displayed ONLY to authorized users, thus apply the strength policy described in Section III C.

After finishing the study, the participants had to fill out a questionnaire (in Appendix 1) so that we can assess their impressions, suggestions and to understand how difficult it was to execute the process (Table 10). Both groups said that the most difficult part was to understand the legal text. The easiest part was to find conflicts. The group that used the process said that the comparison and the extraction were easy, but they had many difficulties to classify and put the text law into the template. Even with this difficulty, they did manage to accomplish the task.

Table 10 Questionnaire summary

Participants	Challenges	Advantages of method
Participant 1	Understand the law text Finding requirements	
Participant 2	Understand the content of the law Time-consuming	
Participant 3	Conflict analysis Verify compliance	Classification helps to discover the requirements Put software requirements in the template
Participant 4	Classify the law requirements Reading law text Identify requirements in the law Transform the text law in requirements	Compare requirements (after putting in the template) Resolve conflict
Participant 5	Law language	Classification helps to extract the requirements and put in the template

The questionnaire included open questions. Participant 4 said: “Sometimes after classification, it is easy and faster to skip the method and define the conflicting or new requirements. The template is necessary with long phrases with a lot of requirements together”. Also, the thesis author followed all the participants doing the task and observed that the three participants that use the method skipped the task of binding terms. When asked, they said that the relationship was very easy and this task was not necessary. Thissuggests that perhaps the method helps in some situations, and some tasks, but in other situations, it can be skipped. We need more studies to analyze if this is an

unnecessary task for all situations, or only in this situation that was a domain known to the participants.

Participant 3 said that an improvement of the method could be “Use more patterns instead of keywords to find the requirements in the law.”

Threats to validity. The objective of this evaluation was to provide a preliminary assessment of the benefits of using this new process. The use of a gold standard for our study lends confidence to the study and limits the risks for construct validity. The gold standard was created for an expert in the method and law to ensure the correctness. Some other factors affecting participant performance during the study are also worth to consider. The skills of the participants involved in the experiments were appropriate to the objective of our preliminary investigation. Moreover, there was no bias of the participants towards the topics covered by the case study.

Conclusion. The effectiveness of the process has been preliminarily evaluated in a case study comparing the proposed process with an ad-hoc one to check requirements compliance. The result of the study concludes that the process can be applied and gives better results than an ad-hoc process. The process helps to extract more requirements, mainly non-functional requirements.

Answering the research questions:

RQ2: What can requirement engineers do to find regulatory requirements from laws in a systematic way? Requirement engineers can use NomosT and the patterns proposed to find the regulatory requirements from the law.

RQ3:What can requirements engineers do to verify software requirements against regulatory requirements in a systematic way? Requirements engineer can put bothsets of requirements in Desiree Templates and use the guidelines to make the comparison.

RQ4:What can requirements engineers do to resolve possible conflicts between regulatory requirements and software requirements in a systematic way? Requirements engineers can weaken, strengthen or drop the software requirements to resolve the possible conflicts.

Desiree ensures the precision of the method since it has a formal form to represent the requirements and to support conflict resolution. The requirements´ discovery in the law text is very dependable on the human performing the task, as such is hard to ensure high precision, since different humans have different skills, although the Nomos notation and the tools help to decrease this human factor and increasing precision.

However, the process needs improvements since it lacks patterns to help mapping the non-functional requirements from text law to the Desiree template. We concluded that the most difficult part is to understand the legal text, so we need better tool support to extract regulatory requirements from the law. One possible solution can be to search for Non-Functional Requirements patterns (Supakkulet al., 2010). Another further direction is improving the automation support for NomosT to Desiree transformation. Also, compliance is a continuous process: the environment where the software is running can change (for example in a mobile application or people accessing data from different countries),or new laws can arise or be updated. In those cases, we have another challenge

that is monitoring this context change that affects the software and laws; thischallengeis showed in the next examples.

5.2
An example of compliance monitor location

In this example, we aim to evaluate the proposal to monitor the changes in location, and how this will affects the compliance. To illustrate our proposal, we will use the same Nursing Scheduler software requirement list used in the previous example (5.1). In this case, we will consider that the software is in the cloud and can run in Italy and Germany. Since the system deals with student´s personal information, the system needs to comply with Privacy Laws.

The Context monitoring is continuous. The software should always be aware of possible changes, so it will always be monitoring the physical environment, for instance, using sensors, such as GPS. If the monitored values show a change, the software needs to adapt.

First it is necessary to identify what are the situations that the software needs to be aware of. To help with this step, we use the Context Awareness Tree (Figure 72). The software located in Germany and the software located in Italy, the time and user in this context do not make a difference because they do not influence the system behavior.

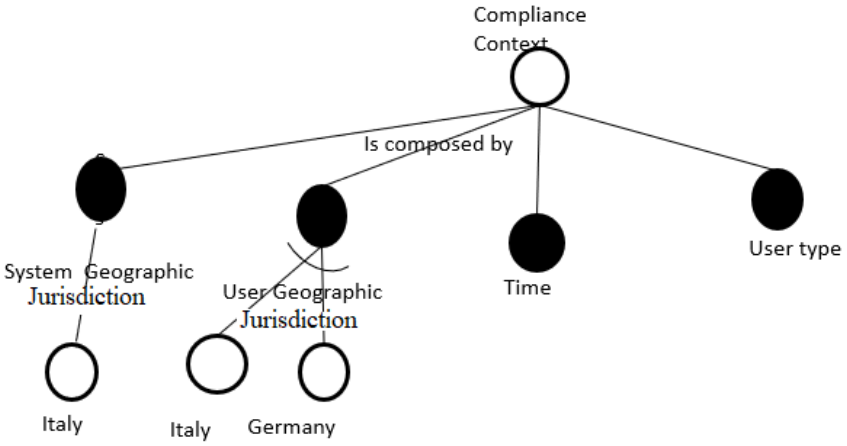

Figure 72Context aware tree for change location

So, we need to have a list of compliant requirements from the Italy Privacy Law(http://www.garanteprivacy.it/home en/Italian-legislation) and from Germany Privacy Law (http://www.gesetze-im-internet.de/englisch_bdsg/). To generate the requirement list the compliance process needs to run for each situation. So, in the end, we will have two SRS lists, one for each country.

Once the SRS is generatedfor each situation, we organize them as a Product Line Requirement Specification Diagram. The software requirements are the same for both situations, some of the software requirements from the Nursing Scheduler software requirement list are:

- *The system shall be capable of processing 100% of nursing students and their classes for the next ten years.*
- *The system shall be expected to manage the nursing program curriculum and class/clinical scheduling for a minimum of 5 years.*
- *Only authorized users shall have access to clinical site information.*
- *Only authorized users shall have access to students' personal information.*
- *Only authorized users shall have access to the portion of the system that interfaces with CampusConnect.*
- *Fit Criterion: Dr. Susan Poslusny and Karen Sysol are the only people who shall have access to the final class section scheduling for the system that interfaces with CampusConnect.*
- *The system shall protect private information from the organization's information policy.*
- *The system shall be able to display a printable summary for individual nursing students, which will include (but not be limited to) student name, student ID, admission date, classes, credits.*
- *The system shall contain contact information (e-mail and phone number) for all people relevant to the system, including (but not limited to) staff members, students, lecture instructors, clinical lab instructors and clinical site administrators.*
- *Program Administrators/Nursing Staff Members shall have the ability to modify information relating to a Nursing Student, including student ID, student name, phone number, e-mail, status (full time or part-him/her as to 1. the identity of the controller, 2. the purposes of collection, processing or use and 3. the categories of recipients.*
- *Personal data stored exclusively for monitoring data protection, safeguarding data or ensuring proper operation of a data processing system may be used exclusively for such purposes.*
- *Personal data which are processed by automated procedures or stored in non-automated filing systems are to be erased if 1. their storage is inadmissible 2. knowledge of them is no longer required by the controller of the filing system for the performance of his duties.*

In our proposal (Engiel, Leite& Mylopolous, 2017) the requirements are represented in Desiree Language (Li et al., 2015). To organize the product line, we also use the Desiree Language, so the requirements from all situations are compared and the requirements that are common for all situations areorganized as the family requirements (PRS)shown in Figure 73. Since all the common part is functional requirement, it is represented by diamonds.

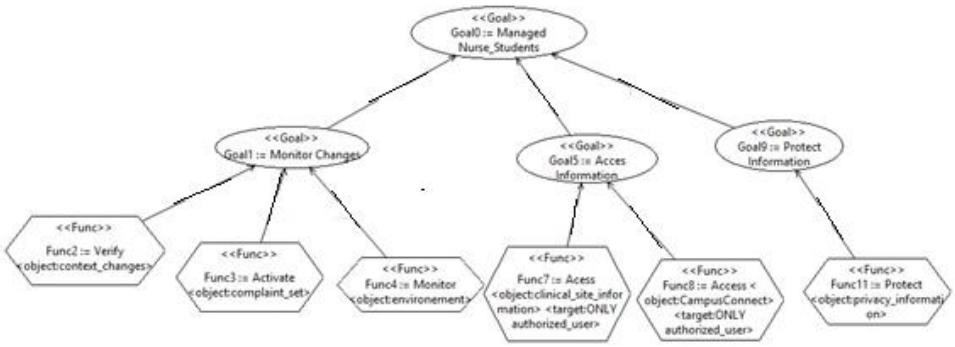

Figure 73 Product line software specification for context monitoring represented in Desiree

The next step is to compare the list of software requirements with the list of regulatory requirements. In this comparison, some new regulatory requirements will arise and need to be addedto the software specification, and some conflicts will be discovered. Each conflict needs to be solved.

For instance, in Germany case, we have a conflict between :

- The software requirements *"The system shall keep personal data for five years"* and
- The regulatory requirement "*Personal data which are processed by automated procedures or stored in non-automated filing systems are to be erased if 1. Their storage is inadmissible 2. Knowledge of them is no longer required by the controller of the filing system for the performance of his duties".*

To resolve this conflict, we need to drop the software requirement since the regulatory requirement is stronger. In this case, the software requirement and the regulatory requirement are opposite, so it is not possible to strengthen or weaken the software requirement, the only solution is to drop it.

The Software Specification for Germany (the requirements that are specific only for Germany) is shown in. The functional requirements are shown in hexagons and constraints as rectangles, for example, the requirement "Display Printable Summary" is represented as a hexagon and the constraint "ONLY with authorization" as a rectangle. These software specifications are after conflict resolution; the C in red represents the conflicts, and the requirements after the black bar are the resolution of the conflict. You can observe that in the lists some software requirement was modified, some requirement was dropped, some continue the same and some new regulatory requirements are added.

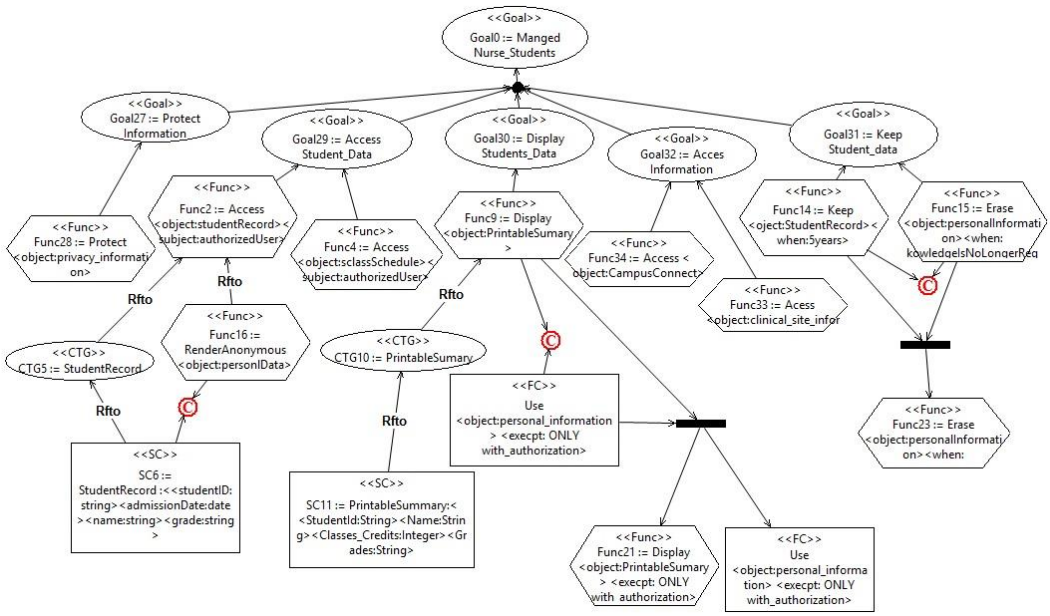

Figure 74 Germany Compliance Specifications

The compliance processes need to run for all the countries that the software can run. Some regulatory requirements discovered in the Italy law are:

- *It shall be prohibited to communicate and disseminate personal data*
- *Data may be: a) destroyed; b) transferred to another controller, provided they are intended for processing which is carried out for purposes similar to those for which they have been collected; c) kept for exclusively personal purposes, without being intended for systematic communication or dissemination;*
- *In respect of the processing of personal data, any data subject shall have the right to: a) be informed, by having access, free of charge, to the register mentioned under paragraph 1, subheading a), of article 31, of the existence of the processing of data that may concern him;*
- *Processing of personal data by private entities or profit-seeking public bodies shall be deemed lawful only if the data subject gives his express consent.*
- *The data subject's consent may relate to the overall processing of one or more of the operations thereof.*

In the same way as Germany, the next step is to verify the conflicts. In the Italian case, it also has a conflict between:

- the software requirement "*The system shall keep personal data for 5 years"* and
- the regulatory requirement "*Data may be: a) destroyed; b) transferred to another controller, provided they are intended for a processing which is carried out for purposes similar to those for which they have been collected; c) kept for exclusively personal purposes, without being intended for systematic communication or dissemination".*

In this case, to resolve the conflict is necessary to modify the software requirement. Putting more information in the requirement, (strengthening) the requirementbecomes compliance, the requirement with modification will be: "The system shall keep personal

data for five years *for exclusively personal purposes, without being intended for systematic communication or dissemination."*
Some software requirements that was not modified or dropped are common for both as:

- *"Only authorized users shall have access to clinical site information,"*
- *"Only authorized users shall have access to the portion of the system that interfaces with CampusConnect"* and
- *"The system shall protect private information from the organization's information policy,"*

The Software Specification for Italy (the requirements that are specific only for)is shown in Figure 75.

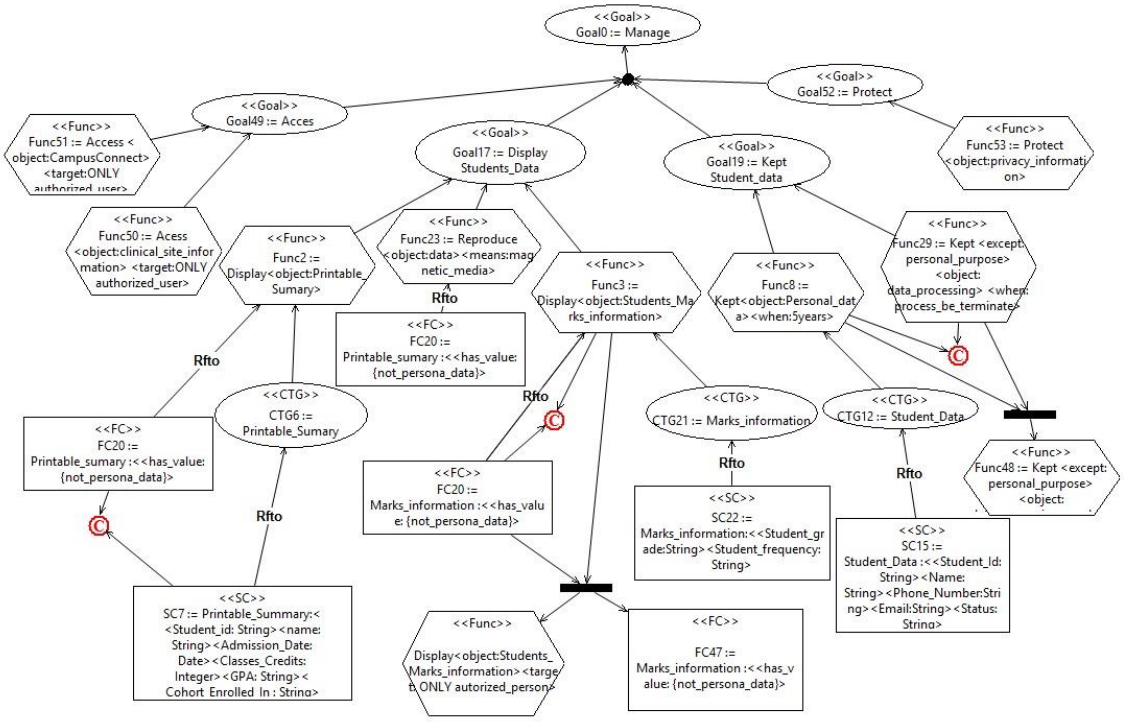

Figure 75 Italy compliance specifications

After both specifications are generated, following the steps of Compliance Process, the software will be running and the monitor activity will start. The Monitor agent will be concerned with the context awareness acquisition, acquiring the data of location and interpreting it: if the data shows that the access is doing from Germany the Germany requirements will be activated, if the data shows that the access is from Italy, the Italy requirements will be activated.

Only the set of requirements that are specific to each situation will change when a change is identified. The set of requirements common to all situation is always running.
Imagine that the system is running in Italy, a nurse student, located in Germany, wants to access your printable summary in the system. The monitor agent will send that the user position is Germany. The Analyzer will verify that the situation change from Italy to Germany and the Planning Agent will find the correct SRS (with Germany regulatory requirements) that need to run. The Executor will change the SRS to the correct one.
So, when the nurse student access the Printable Summary, he will access all the data, but the data will be rendered anonymous, so the student needs to know his id, to know his information (Figure 76).

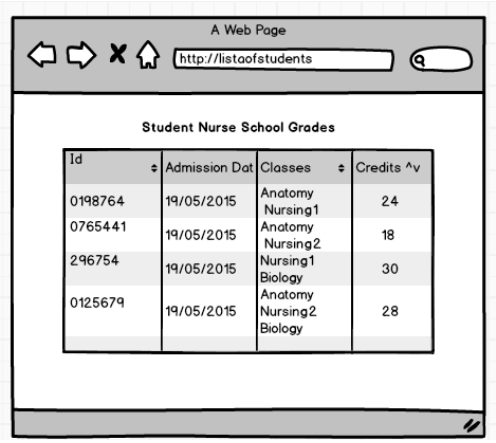

Figure 76 Germany Student Printable Summary

However, if the nurse student is in Italy and tries to access the system, the Monitor agent will acquire the data and will send that the user position is Italy. The Analyzer will verify that the situation change from Germany to Italy and the Planning Agent will find the correct SRS (with Italy regulatory requirements) that need to run. The Executor will change the SRS to the correct one. In this case, the printable summary can be accessed only by the student, sothe system will show only his data (Figure 77).

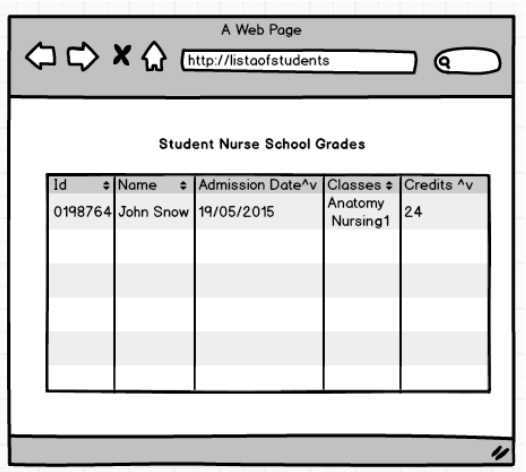

Figure 77 Italian Student Printable Summary

The example shows how it is important to deal with a different set of requirements for each jurisdiction that the software is running in. In the example, we could identify differences that can deal with compliance problems.

5.3
Car example of context change

To illustrate our proposal and answer the research question RQ5:**What can requirements engineers do to ensure that the software is compliant over time?** We will use the example of a company that rents cars. The company is located in Washington and Canada. People can pick the car in Canada and return to Washington or the other way

if the case is, the car is a level one of a self-driving car, where drivers and automated system shares control over the vehicle, so the driver control steering and the automated system control, e.g., the speed and car distance. For this example, we will consider that car is crossing the border from the USA (Washington) and Canada. Two people will split the direction, one is 24 years old and the other is 16 years old. Since the system deals with level 1 of autonomous car and traffic, the system needs to be compliant with Traffic Laws. The autonomous car is still a novelty, so there is no specific legislation.

To answer the question, we evaluated the quality of the output. We considered that better quality output entails greater compliance.Weused the measure of the number of changes that is necessary to adapt to the different situations.

(a) Materials: The example was conducted using a data set of DARPA Urban Challenge(http://archive.darpa.mil/grandchallenge/docs/Urban_Challenge_Rules_102707.pd) that have the software requirements along with part RCW 46.61.400 of Washington Traffic Law (http://apps.leg.wa.gov/rcw/default.aspx?cite=46.61), since in the USA each state has its legislation and Part Ix Rate of Speed from Canada traffic law (https://www.ontario.ca/laws/statute/90h08). It is necessary to remember that the example is about level 1 of autonomous car, so the drivers share the control with the car.

(b) Participants: the author of this thesis that hasten years of experience in requirement engineer (academia and business) and five years of experience dealing with law and requirements (academia)

(c) Tasks: The Participants had first to discover all the situations that the car can meet and then find requirements in the laws and compare them with the pre-existing requirements, checking for conflicts. If conflicts werefound, the participants needed to resolve them. With all the requirements is necessary to generate the different SRS for each situation.

The first step is to identify what are the situations that the software needs to be aware. For this, we use the compliance instance tree Figure 78. In this case for the Geographic Jurisdiction we have two situations USA –Washington to Canada, time can be day or night, and for the user, over 16 or under 16, since it is the minimum age to drive in the USA.

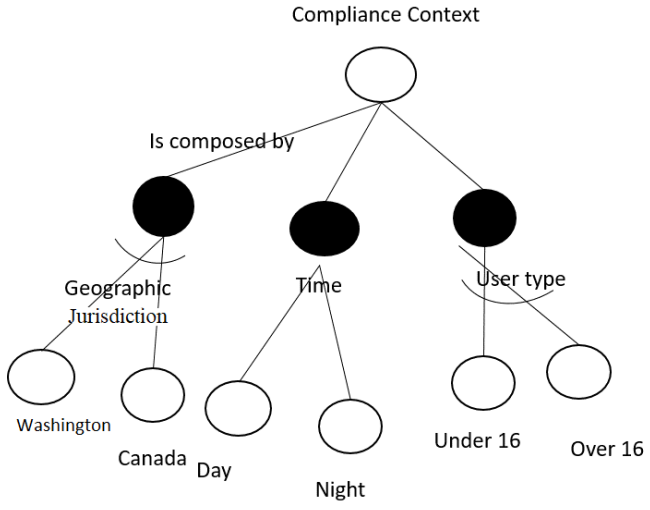

Figure 78Compliance instance tree

So, in this case,there are eight situations which are the combinations of the changes in the country (USA –Washington to Canada), time (day or night) and user (over 16 or under 16). To represent all the possible changes, the situations are represented in a state diagram, Figure 79:

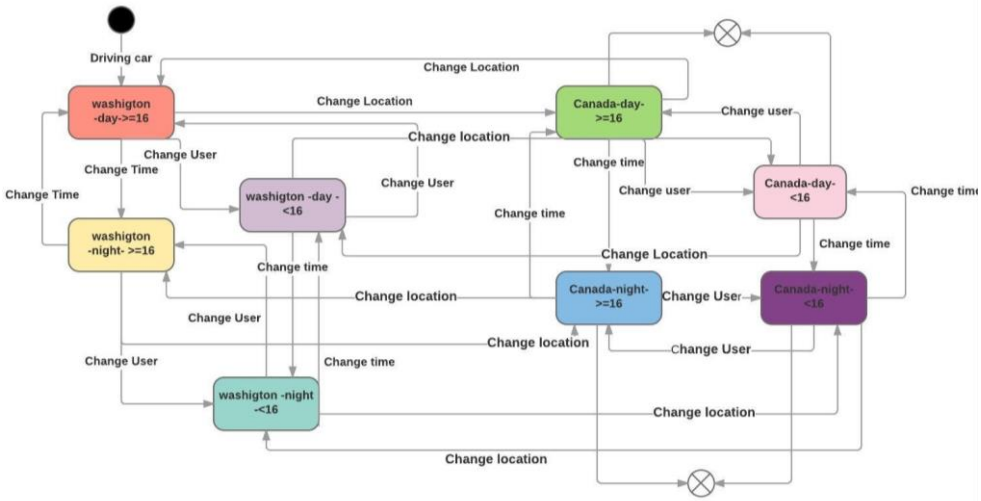

Figure 79 State diagram for the situation change

Therefore, it is necessary to run the Compliance Process for Canada and Washington law to generate the SRS for each situation. The steps necessary are:

(1) Discover the regulatory requirements in the law (tool supported by NomosT (Zeni et al. 2016))

(2) Compare with the software requirements (tool supported by Desiree) and

(3) Resolve the possible conflicts.

The software requirements presented in DARPA Urban Challenge are the same for both situations, those requirements are used as the need of the automatous car, how the automatous car needs to work:

- Exhibit context-dependent speed control to ensure safe following distance.
- Exhibit safe check-and-go behavior when pulling around a stopped vehicle, pulling out of parking spot, moving through intersections, and in situations where a collision is possible.
- Stay on the road and in a legal and appropriate travel lane while en route operation, including adherence to speed limits.
- Exhibit safe-following behavior when approaching other vehicles from behind in a traffic lane. This includes maintaining a safe-distance
- Navigate safely in areas where GPS signals are partially or entirely blocked.
- Follow paved and unpaved roads and stay in lane with very sparse or low accuracy GPS waypoints.
- Change lanes safely when legal and appropriate, such as when passing a vehicle or entering an opposing traffic lane to pass a stopped vehicle. Vehicles must not pass other vehicles queued at an intersection.
- Merge safely with traffic moving in one or more lanes after stopping at an intersection.
- Pull across one lane of moving traffic to merge with moving traffic in the opposing lane.

- Stop safely within 1 meter of the stop line at a stop sign intersection and proceed without excessive delay (less than 10 seconds) according to intersection precedence rules.
- Exhibit proper queue behavior at an intersection, including stopping at a safe distance from other vehicles and stop-and-go procession to the stop line without excessive delay.
- Navigate toward a destination in a large, open area where minimal or no GPS points are provided, as in loading dock areas or parking lots. These areas may contain fixed obstacles such as parked vehicles and moving obstacles including other vehicles.
- Safely pull into and back out of a specified parking space in a parking lot.
- Dynamically re-plan and execute the route to a destination if the primary route is blocked or impassable. The following behaviors or capabilities are outside the scope of this program:
- Recognition of external traffic signals such as traffic lights and stop signs, through the use of sensors.
- Recognition of pedestrians and pedestrian avoidance.
- Behaviors necessary for highway driving such as high-speed passing or high speed merge at an onramp. Speed limits for the Urban Challenge will be 30 mph or less
- Speed limits: Vehicle speed conforms to minimum and maximum limits.
- Collisions: Vehicle acts to avoid collisions and near-collisions at all times, as judged by DARPA.
- Stop line: Vehicle stops, so front bumper is within 1 meter of stop line at the intersection
- Leaving lane to pass: Vehicle maintains a forward vehicle separation of one vehicle length when leaving the lane to initiate a passing maneuver in a travel area.
- Returning to lane after pass: Vehicle returns to travel lane between one and four vehicle lengths when completing a passing maneuver.
- Vehicle respects precedence order at intersections and does not proceed out of turn.
- Minimum following distance: When following a moving traffic-vehicle, the autonomous vehicle maintains the required forward vehicle separation distance.
- Queueing: Vehicle exhibits correct stop-and-go queueing behavior in a line of stopped vehicles, always maintaining a minimum spacing equal to the forward vehicle separation distance and a maximum spacing of two vehicle lengths.
- Obstacle field: Vehicle demonstrates the ability to negotiate an obstacle field safely and effectively.
- Parking lot: Vehicle exhibits correct parking lot behavior and demonstrates the ability to pull forward into and reverse out of a specified parking spot.
- Dynamic re-planning: Vehicle exhibits behaviors necessary to achieve objective checkpoints when roads are blocked.
- Road following: Vehicle navigates roads with sparse waypoints and stays in travel lane through road-following by sensing berms or road edges, or by any other sensor-based technique.
- Emergency braking: Vehicle comes to a complete and safe stop to avoid a collision when a moving obstacle suddenly moves into the travel lane.

- Traffic jam: When encountering a partially-blocked intersection, vehicle maneuvers to make forward progress and avoids collisions

The Compliance Process run for the Canadian law some regulatory requirements discovered in law are:

- No person shall drive a motor vehicle at a rate of speed greater than50 kilometers per hour on a highway within a local municipality or a built-up area;
- 80 kilometres per hour on a highway, not within a built-up area, that is within a local municipality that had the status of a township on December 31, 2002, and, but for the enactment of the *Municipal Act, 2001*, would have had the status of a township on January 1, 2003, if the municipality is prescribed by regulation;
- 80 kilometers per hour on a highway designated by the Lieutenant Governor in Council as a controlled-access highway under the *Public Transportation and Highway Improvement Act*, whether or not the highway is within a local municipality or built-up area;
- The council of a municipality may, for motor vehicles driven on a highway or portion of a highway under its jurisdiction, by by-law prescribe a rate of speed different from the rate set out in subsection (1) that is not greater than 100 kilometres per hour and may prescribe different rates of speed for different times of day. 2006, c. 32, Sched. D, s. 4 (3).
- The rate of speed set under subsection (10) may be any speed that is not greater than 100 kilometers per hour. 2006, c. 32, Sched. D, s. 4 (4).

The next step is to compare the list of software requirements with the list of regulatory requirements. In this comparison, some new regulatory requirements will arise and need to be addedto the software specifications, and some conflicts were discovered. Each conflict needs to be solved.

For instance in Canadian law, some conflicts discovered are:

- Software requirements: “Behaviors necessary for highway driving such as high-speed passing or high speed merge at an onramp. Speed limits for the Urban Challenge will be 30 mph or less.”
- The regulatory requirements: “ No person shall be a motor vehicle at a rate of speed greater than50 kilometers per hour on a highway within a local municipality or a built-up area”;

This case was identified as a conflict since 30mph is equivalent to 48km/h. So, the car will have the high speed of 48km/h, and the legislation allows 50 km/h.

Part of the Canada Software Requirement Specification model in Desiree Language (Li et al., 2015) is shown in Figure 80, in Desiree language the requirements are connected to one goal, for this was necessary to create goals to organize the requirements, the goal of the traffic law is to obey the traffic law and it can be split in three other goals, Respect speed limit, Change speed limit and Penalty rule violation.

The Function Drive motor vehicle in built-up are represented as Func 5: Drive <object:motor_vehicle><where: builtl_up_are> and have 2 quality constrain that are conflicting (mark as C in red) QC: Limit rate < Limit <50Km/H> QC: Limit rate <Limit <48 Km>. The requirement are related to the goal Respect max limit allowed that is related to the goal of Respect speed limit. The quality constrains CQ Limit rate < Limit

<50Km/H> was dropped, so after the black bar, it is represented only the QC: Limit rate <Limit <48 Km> after the conflict resolution.

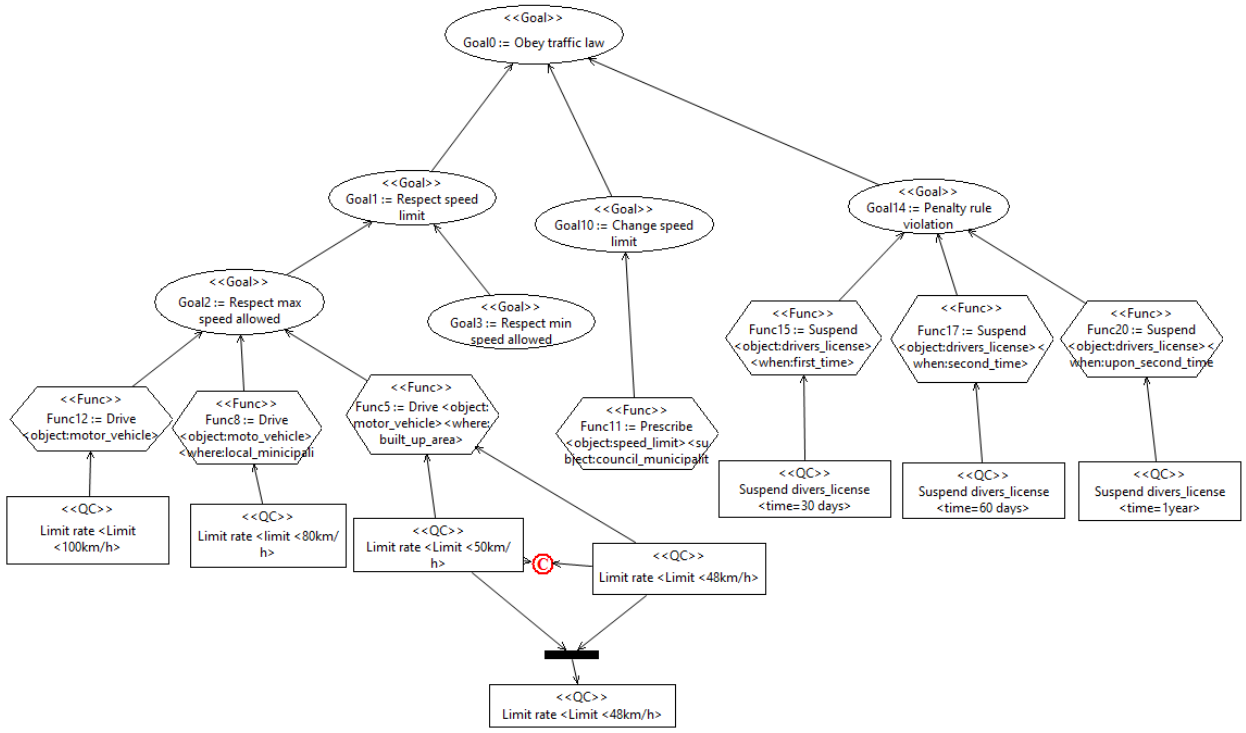

Figure 80 Canada software specification

The compliance processes need to run for all the countries that the software can run. Some regulatory requirements discovered in the Washington law are:

- No person shall drive a vehicle on a highway at a speed greater than it is reasonable and prudent under the conditions and having regard to the actual and potential hazards then existing.
- In every event speed shall be so controlled as may be necessary to avoid colliding with any person, vehicle or other conveyance on or entering the highway in compliance with legal requirements and the duty of all persons to use due care.
- Except when a special hazard exists that requires lower speed for compliance with subsection (1) of this section, the limits specified in this section or established as from now on authorized shall be maximum lawful speeds, and no person shall drive a vehicle on a highway at speedmore than such maximum limits. (a) Twenty-five miles per hour on city and town streets;(b) Fifty miles per hour on county roads;(c) Sixty miles per hour on state highways.

The next step is to verify the conflicts. We have a conflict between:

- Software requirement *"*Behaviors necessary for highway driving such as high-speed passing or high speed merge at an onramp. Speed limits for the Urban Challenge will be 30 mph or less."
- The *r*egulatory requirement "no person shall drive a vehicle on a highway at speedmore than such maximum limits. (a) Twenty-five miles per hour on city and town streets;"

In this case, we need to drop the software requirement, since the regulatory requirement is stronger than the software requirement. For both sets of requirements being in the same representation, to build the product line, the mph was converted the km/h.

Part of Software Specification for Washington is shown in Figure 81. The Function Drive motor vehicle in built-up are represented as Func 5: Drive <object:motor_vehicle><where: builtl_up_are> and have 2 quality constrains that are conflicting (mark as C in red) QC: Limit rate < Limit <40Km/H> QC: Limit rate <Limit <48 Km>. The quality constrains CQ Limit rate < Limit <48Km/H>was dropped, so after the black bar, is represented only the QC: Limit rate <Limit <40 Km> after the conflict resolution.

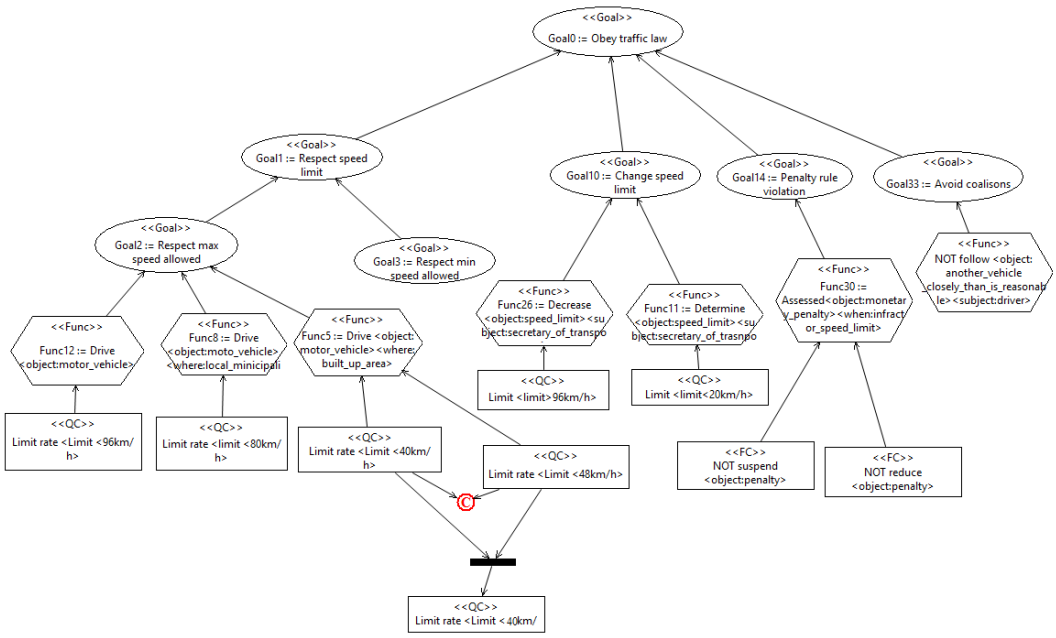

Figure 81Washington software specification

After both specifications are generated, we have the list of compliance requirements that need to be followed. The next step is to organize those requirements as the product line. For this, it is necessary to generate the SRS for each situation identified. The situations are the combination of user, time and location, and we have 8 situations:

1. Washington, day, over 16
2. Washington, night, over 16
3. Washington, night, under 16
4. Washington, day, under 16
5. Canada, day, over 16

6. Canada, day, under 16
7. Canada, night, over 16
8. Canada, night, under 16

It is necessary to see the compliance requirements from Canada and Washington (after the compliance process) and classify which requirement is related to each situation. Some software requirements are common for all situations. So, they will be in the product line requirement specification, they are:

- Exhibit context-dependent speed control to ensure safe operation, including adherence to speed limits.
  - Exhibit <object: context-dependent speed control>
  - Ensure<object: Safe-operation>
  - Adhere<object: speed_limits>
- Maintaining a safe-following distance.
  - Maintaining <object:safe-following distance>
- Navigate safely in areas where GPS signals are partially or entirely blocked.
  - Navigate (atomonous_car)::safely
- Follow paved and unpaved roads and stay in lane with very sparse or low accuracy GPS waypoints.
  - Follow <object:pavedRoads>
  - Follow <object:unpavedroads>
- Change lanes safely when legal and appropriate, such as when passing a vehicle or entering an opposing traffic lane to pass a stopped vehicle. Vehicles must not pass other vehicles queued at an intersection.
  - NOT pass <object:other vehicles><when:queued at an intersection>
  - Change (line)::safely
- Stop safely within 1 meter of the stop line at a stop sign intersection and proceed without excessive delay (less than 10 seconds) according to intersection precedence rules.
  - Stop < object: within 1 meter of the stop lime at stop sign intersection>
- Exhibit proper queue behavior at an intersection, including stopping at a safe distance from other vehicles and stop-and-go procession to the stop line without excessive delay.
  - Stop <object: safes distance from other vehicles>
- Recognition of external traffic signals such as traffic lights and stop signs, through the use of sensors.
  - Recognize<object: trafficsignals>
  - Use <object:sensors>
- Recognition of pedestrians and pedestrian avoidance.
  - Recognize<object:pedestrian>
- Speed limits: Vehicle speed conforms to minimum and maximum limits.
  - Conforms <object:speed to minimum limits>
  - Conforms<object:speed to maximum limits>
- Minimum following distance: When following a moving traffic-vehicle, the autonomous vehicle maintains the required forward vehicle separation distance.
  - Maintains <object: required forward vehicle separation>
- Dynamic re-planning: Vehicle exhibits behaviors necessary to achieve objective checkpoints when roads are blocked.
  - Re-planning<object:routes><when: roads are blocked>

The other will be in the specific software requirement specification, to help to organize the requirements for each situation (the product line) Table 11 was created.

Table 6Specification for each situation

#	Location	Time	User
1	FC:=Determine <subject:secretary of transportation><object: maximium speed hereinbefore set forth is greater than is reasonable or safe with respect to a state highway under the conditions found> FC:=Assessed <object: a monetary penalty equal to twice the valor of the penalty><when: have committed any infraction relating to speed restriction within a school or playground speed zone> QC:=NOT suspend <object:penalty> QC:=NOT reduce <object:penalty> QC:=NOT waive <object:penalty> FC:=Decrease <object : limit><subject: local authorities> QC:=Limit <limit: >60miles/h> FC:=Increases <object : limit><subject: local authorities> QC:=Limit rate <limit: <20miles/h> QC:=NOT follow <object: another vehicle more closely than is reasonable><subject:driver of a motor vehicle> FC:=Drive <subject:no one><when: alcohol concentrarion of 0.08> QC:=Limit <limit: >60miles/h> -96 km	FC:=Turn off <object:headlights>< when: 30 minutes before sunset ><subject:drivers>	FC:=Drive <subject:no person under 16><object:motor vehicle>
2	FC:=Determine <subject:secretary of transportation><object: maximium speed hereinbefore set forth is greater than is reasonable or safe with respect to a state highway under the conditions found> FC:=Assessed <object: a monetary penalty equal to twice the valor of the penalty><when: have committed any infraction relating to speed restriction within a school or playground speed zone> QC:=NOT suspend <object:penalty> QC:=NOT reduce <object:penalty> QC:=NOT waive <object:penalty> FC:=Decrease <object : limit><subject: local authorities> QC:=Limit <limit: >60miles/h> FC:=Increases <object : limit><subject: local authorities> QC:=Limit rate <limit: <20miles/h> QC:=NOT follow <object: another vehicle more closely than is reasonable><subject:driver of a motor vehicle> FC:=Drive <subject:no one><when: alcohol concentrarion of 0.08> Limit <limit: >60miles/h> -96 km	FC:=Turn on <object:headlights>< when: visibility is low for any reason><subject:drivers> FC:=Turn on <object:headlights>< when: 30 minutes after sunset ><subject:drivers>	FC:=Drive <subject:no person under 16><object:motor vehicle>

#	Location	Time	User
3	FC:=Determine <subject:secretary of transportation><object: maximium speed hereinbefore set forth is greater than is reasonable or safe with respect to a state highway under the conditions found> FC:=Assessed <object: a monetary penalty equal to twice the valor of the penalty><when: have committed any infraction relating to speed restriction within a school or playground speed zone> QC:=NOT suspend <object:penalty> QC:=NOT reduce <object:penalty> QC:=NOT waive <object:penalty> FC:=Decrease <object : limit><subject: local authorities> QC:=Limit <limit: >60miles/h> FC:=Increases <object : limit><subject: local authorities> QC:=Limit rate <limit: <20miles/h> QC:=NOT follow <object: another vehicle more closely than is reasonable><subject:driver of a motor vehicle> FC:=Drive <subject:no one><when: alcohol concentrarion of 0.08> Limit <limit: >60miles/h> -96 km	FC:=Turn off <object:headlights>< when: 30 minutes before sunset ><subject:drivers>	FC:=Drive <subject:no person under 16><object:motor vehicle>
4	FC:=Determine <subject:secretary of transportation><object: maximium speed hereinbefore set forth is greater than is reasonable or safe with respect to a state highway under the conditions found> FC:=Assessed <object: a monetary penalty equal to twice the valor of the penalty><when: have committed any infraction relating to speed restriction within a school or playground speed zone> QC:=NOT suspend <object:penalty> QC:=NOT reduce <object:penalty> QC:=NOT waive <object:penalty> FC:=Decrease <object : limit><subject: local authorities> QC:=Limit <limit: >60miles/h> FC:=Increases <object : limit><subject: local authorities> QC:=Limit rate <limit: <20miles/h> NOT follow <object: another vehicle more closely than is reasonable><subject:driver of a motor vehicle> FC:=Drive <subject:no one><when: alcohol concentrarion of 0.08> Limit <limit: >60miles/h> -96 km	FC:=Turn on <object:headlights>< when: visibility is low for any reason><subject:drivers> FC:=Turn on <object:headlights>< when: 30 minutes after sunset ><subject:drivers>	FC:=Drive <subject:no person under 16><object:motor vehicle>
5	FC:=Prescribe <subject: the council of municipality><object: for motor vehicle driven><where: on highway or portion of a highway under its jurisdiction>< object: rate of speed different from the rate set out>	FC:=Turn off<object:headlights>	FC:=Drive <subject:no person under 16><object:motor vehicle>

#	Location	Time	User
	FC:=Prescribe <subject: the council of municipality><object: for motor vehicle driven><where: travelling down grade on that portion of the highway >< object: a lower rate of speed> FC:=Set <object: lower rate of speed><target: to motor vehicles driven><where: in the designated construction zone ><subject:person authorized> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit> FC:=Suspend the driver´s license <tim: > 30 days> FC:=Suspend the driver´s license < time: > 60 days> Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit upon the first subsequent conviction > FC:=Suspend the driver´s license < time: > 1 year> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit upon the second subsequent conviction > FC:=Be driven <Object: no motor vehicle><where: on a highway at such a slow rate of speed><except: when the slow rate is necessary for safe operations> FC:=Surrender <object:driver´s license><when: blood is 50miligrams or more of Alcohol in 100mililitres> Speed rate <speed: <100km>		
6	FC:=Prescribe <subject: the council of municipality><object: for motor vehicle driven><where: on highway or portion of a highway under its jurisdiction>< object: rate of speed different from the rate set out> FC:=Prescribe <subject: the council of municipality><object: for motor vehicle driven><where: travelling down grade on that portion of the highway >< object: a lower rate of speed> FC:=Set <object: lower rate of speed><target: to motor vehicles driven><where: in the designated construction zone ><subject:person authorized> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit> FC:=Suspend the driver´s license <tim: > 30 days> FC:=Suspend the driver´s license < time: > 60 days> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit upon the first subsequent conviction >	FC:=Turn on<object:headlights><when:visibility is low>	FC:=Drive <subject:no person under 16><object:motor vehicle>

#	Location	Time	User
	FC:=Suspend the driver´s license < time: > 1 year> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit upon the second subsequent conviction > FC:=Be driven <Object: no motor vehicle><where: on a highway at such a slow rate of speed><except: when the slow rate is necessary for safe operations> FC:=Surrender <object:driver´s license><when: blood is 50miligrams or more of Alcohol in 100mililitres> Speed rate <speed: <100km>		
7	FC:=Prescribe <subject: the council of municipality><object: for motor vehicle driven><where: on highway or portion of a highway under its jurisdiction>< object: rate of speed different from the rate set out> FC:=Prescribe <subject: the council of municipality><object: for motor vehicle driven><where: travelling down grade on that portion of the highway >< object: a lower rate of speed> FC:=Set <object: lower rate of speed><target: to motor vehicles driven><where: in the designated construction zone ><subject:person authorized> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit> FC:=Suspend the driver´s license <tim: > 30 days> FC:=Suspend the driver´s license < time: > 60 days> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit upon the first subsequent conviction > FC:=Suspend the driver´s license < time: > 1 year> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit upon the second subsequent conviction > FC:=Be driven <Object: no motor vehicle><where: on a highway at such a slow rate of speed><except: when the slow rate is necessary for safe operations> FC:=Surrender <object:driver´s license><when: blood is 50miligrams or more of Alcohol in 100mililitres> Speed rate <speed: <100km>	FC:=Turn off<object:headlights>	FC:=Drive <subject:no person under 16><object:motor vehicle><exception:have a supervisor and is over14>
8	FC:=Prescribe <subject: the council of municipality><object: for motor vehicle driven><where: on highway or portion of a highway under its jurisdiction>< object: rate of speed different from the rate set out>	FC:=Turn on<object:headlights><when:visibility is low>	FC:=Drive <subject:no person under 16><object:motor vehicle><exception:have a supervisor and is over14>

#	Location	Time	User
	FC:=Prescribe <subject: the council of municipality><object: for motor vehicle driven><where: travelling down grade on that portion of the highway >< object: a lower rate of speed> FC:=Set <object: lower rate of speed><target: to motor vehicles driven><where: in the designated construction zone ><subject:person authorized> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit> FC:=Suspend the driver´s license <tim: > 30 days> FC:=Suspend the driver´s license < time: > 60 days> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit upon the first subsequent conviction > FC:=Suspend the driver´s license < time: > 1 year> FC:=Suspend <object: driver´s license><target:person driving at a rate of sped of 50 or more lm greater that the speed limit upon the second subsequent conviction > FC:=Be driven <Object: no motor vehicle><where: on a highway at such a slow rate of speed><except: when the slow rate is necessary for safe operations> FC:=Surrender <object:driver´s license><when: blood is 50miligrams or more of Alcohol in 100mililitres> Speed rate <speed: <100km>		

It is also necessary to verify if there are requirements for the different legislation that conflicts with it. Some regulatory requirement from Canada can conflict with the regulatory requirement from Washington; those situations can generate penalties and charges if the system does not adapt to the correct situation. There were found four requirements that conflict with each other:

- Washington: Surrender <object:driver´s license><when: blood is 50miligrams or more of Alcohol in 100mililitres>
- Canada: Drive <subject:no one><when: alcohol concentration of 0.08>

- Washington: Speed rate <speed: <100km>
- Canada: Limit <limit: >60miles/h>

In the case study, it was found 15 changes in requirements for the different situations in the product line, four of them conflict between them, and that can generate penalties. The effort in finding conflict requirement between regulatory requirements and software requirement is the same as the effort to create the product line. In both cases it´s necessary to compare both lists of requirements and see the difference between them.

To illustrate all the examples: The 24 years old is driving the car, during the day in Washington at a speed of 25 miles (the limit speed) – Situation 1 in Product Line, when crossing the border. The system needs to be aware of the change of location. The monitor will send the signal to the satellite that will return the data, the system will calculate the position and verify that position changes from Canada to Washington, and the system needs to change the SRS from situation 1 to the SRS related to the situation older than 16, day and Washington – Situation 5.When SRS changes, the system can increase the speed rate to 30 miles and will stay compliant.

In the same way, the software will monitor the date and from time to time will pick the system date. Now the 24 years older is driving in Canada, and the system verifies that the time is during the sunset, so the analyzer will realize that the system needs to change from situation 5 to situation 6 and activate the correct SRS: Washington, night and older 16, and will turn on the headlights to be compliant. The context change is shown in a state diagram (Figure 79)

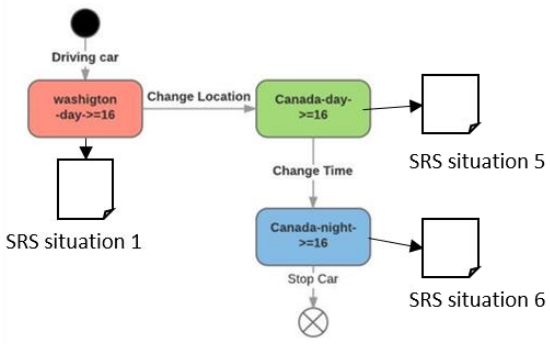

Figure 79 State diagram for the running example

Threats to validity. The objective of this example was to provide a preliminary assessment of the use of monitoring context. External validity of our study is concerned with the generalization of the results to other cases and using real cases. We consider the results preliminary, so they need to be compared with other experiments including to use other participants. One experiment using the complete text of the laws and an industrial size project is very important to check the performance of the process.

Conclusions. We can observe that the context have a strong influence on the software requirements. If there is no monitoring of the environment changes, and mechanisms to ensure that the software is always compliant, it may cause non-conformity problems and may incur penalties, including financial ones, for example, the fine for the headlights is $130, and depending on the country the time to turn it on is completely different. We also can answer the **RQ5: What can requirements engineers do to ensure that the software is compliant over time?** Requirement engineers can use Compliance Monitor to ensure that the software is compliant over time.

5.4 Chapter Summary

In this Chapter, we present the results of the evaluations where we can analyze the use of the proposed framework. We did three evaluations: one only for the compliance process, the one to evaluate the monitor of the location change and the last one to evaluate the entire proposed framework. The evaluation shows us some improvements that need to be done in the framework as creating patterns to discover for the non-functional requirement, an automatic way to transform Nomos language to Desiree language and an implementation of the agents. We also showed the viability of the idea of continuous compliance, consisting of a compliance process and a compliance monitoring. It is necessary to do a better evaluation in a large case study, using the framework proposed and an ad-hoc method comparing the results.

6
Conclusion

This chapter presents the conclusions of the study. The research contributions, the limitations and suggests future work.

All organizations are subject to the law. The demand for regulatory compliance is growing up and consequently affecting the design of software system. The organizations need to ensure that the software functionalities developed are compliant with the legislation. This work shows that enforcing compliance is an error-prone and time-consuming process that requires legal knowledge as well as requirements engineering skills. Organizations need methods and tools to help them to ensure compliance.

This work proposes the compliance process that supports the extraction of compliance requirements, comparing them with software requirements and resolving conflicts that may arise. Some tools and frameworks support the process. The work also proposes a set of heuristics that help to connect the different frameworks used in this paper. The heuristics aim to support the tasks that are still manually. We believe that with a guideline, it becomes easier for the requirements engineer to execute them.

To build software compliant with laws first is necessary to discover the legal requirements that software needs to respect. So, it is necessary to discover the laws that are the source of information and then elicit the requirements that affect the software system. This task is a challenge because laws are voluminous and requirements engineers have difficulty with the language that laws are written. This work proposes the use of NomosT framework to extract the requirements from the law. The tool uses using semantic parametrization, was created some patterns that aim to represent the duties and rights to help the extraction.

After discovering the regulatory requirements is necessary to verify if the software requirements that already exist in the organization have some conflicts with them. The problem is the gap between legal language and requirement language that makes hard to compare both sets of the requirement to understand the nonconformities. Requirements come from stakeholders needs and are written in a specific form. Laws describe what the subjects which are addressed by the law can or cannot, and the laws were written by lawyers or law specialists with other formalism. In this point, the Eunomia proposes the use of Eunomia Framework to compare the regulatory requirements with software requirement and resolve the possible conflicts. First is necessary to put bothset of requirements in some terminology, the thesis proposes some patterns and guideline to help in this point.

The heuristics aim to support the tasks that are still manually. We believe that with a guideline, it becomes easier for the requirements engineer to execute them. To resolve the conflicts, the requirements are dropped, weaken or strengthen.

In requirement engineer, another critical phase is to manage the requirements. Laws and software requirements are not static and can change, and the system needs to stay updated and consistent. Recent paradigms, such as cloud computing, and mobiles, also require software to operate in different environments and can need to comply with a different jurisdiction. In these environments, compliance requirements can vary at runtime and traditional compliance management techniques may no longer be sufficient.

To deal with the challenge of changes, this work proposes the Continuous Compliance, a method that combines context awareness, agents and the software product line. The agents act to ensure that the system is always compliant, they monitor the environment, and when a change is observed that act to ensure that the correct requirement specification will be running. The product line organizes all the requirement specification based on the legislation that the software must compliant.

6.1. Contributions

The main contribution of Eunomia Framework is to address the challenges of continuous software compliance with process and goal model perspective with emphasis on monitoring. This process equips the requirements engineering team with a systematic method helped by tool support. The process (Engiel, Leite & Myloupolos, 2017) address the challenging of eliciting the regulatory requirement from the law, compare them with software requirement, putting them in the same representation and the conflict resolution. Some steps are tool supported, and the manual ones have a heuristic to help to execute the task.

The process aims to reduce the time effort. The laws usually have hundreds of pages, to deal with them, is very time-consuming. Without a tool it is necessary to read all the law to find the requirements, in this proposal using NomosT (Zeni et al. 2016) (Zeni et al., 2018) the tool leads with the hundreds of pages in seconds, and the process helps to create patterns and heuristics to parameterize a better automatic search.

Also, the process aims to elaborate a consistent requirement specification. The requirement specification generated for the process has the software requirements, the regulatory requirements and the software requirement modified to be compliant with the regulatory requirement. The process supports the translation of regulatory requirement to software requirements and allows the comparison, having heuristics to guide. The Desiree tool supports all the traceability showing graphically the requirements modified or dropped.

In the case study, was observed that the big challenge for the requirement engineer was to deal with the law text. The process helps to build the bridge between

law and requirement engineers once the requirement engineer does not need to understand or read the law text. The NomosT tool leads with the law and returns an annotated text that the requirement engineer is used to working with, and the heuristics help to transform this annotated text into a requirement specification in Desiree language. The law expert presence continues to be necessary, in the moments of defining the keywords, mapping legal terms and in conflict resolution, but it is also not necessary for him to know how to deal with requirements specification.

The work also contributes to higher transparency, since it creates a complete trace of legal norms and requirements. Tracing is done primarily in the law, marking where legal requirements exist, then between legal requirements which are new and which have a relationship to software requirements and last which software requirements have been modified or removed to ensure compliance.

Another significant contribution was the creation of compliance context tree, showing which context changes influence legislation and thus compliance. With this mapping, it becomes easier to understand the impacts of context change in compliance.

As well this work concern not only with changes in legislation and software requirements but also to changes in the environment. Monitoring the environment with an adaptation of systems in real time to ensure compliance was also a contribution to this work. The modeling of the MAPE agents as well as the creation of the i * and BPMN specification of this behavior helps the development and programming of these agents.

The effectiveness of the compliance process has evaluated in a case study comparing the proposed process with an ad-hoc one to check requirements compliance. The result of the study concludes that the process can be applied and gives better results than an ad-hoc process. The process helps to extract more requirements, mainly non-functional requirements.

The examples of compliance monitoring show how is essential to deal with a different set of requirements. For each jurisdiction that the software is running, there are the different context that needs to be considered. In the example, we could identify differences that can deal with compliance problems and may lead to penalties. The example shows that is possible to monitor the changes and have a set of requirements for each situation that need to be activated. Also, it is possible to have mechanisms to change for the right set of requirements in runtime.

Our proposal differs from others in the literature in that it consists of a tool-supported compliance process that addresses the identification, extraction, comparison and conflict resolution to establish compliance. Also, our proposal is the concern with the continuous compliance where the system needs to adapt to changes to maintain the compliance. The other works lead only with one part of the process and are not concerned with the continuous process and all the changes that can happen.

6.2.
Limitations

The first limitation of Eunomia is that the requirements discovery in the law text is very dependable of the human performing the task, as such, it is difficult to promote a high precision since different humans have different skills. To minimize this limitation, the use of Nomos notation, NomosT tool, and Desiree helps to decrease this human factor and increasing precision. Desiree promotes the precision of the compliance process since it has a formal form to represent requirements and to support conflict resolution. NomosT promotes better precision of the regulatory requirement extraction because can have patterns and heuristics, and the tool will search for this heuristic, the precision of a tool is higher than the human eye, even more with hundreds of pages.

Anotherlimitation is that the patterns are created only to English languages, so in this version of the framework we can only extract requirements of the law written in English. Some countries even English not being the mother languages have their laws translated to it, like Italy and Germany. Although to use the framework in other languages is necessary to verify the structure of the written, the patterns and configurethe NomoT for this new context. Also, the patterns to translate the regulatory requirements to Desiree language need to be adapted to the other language.

Eunomia does not work for all types of norms and all requirement specification. Eunomia uses laws that are represented in explicit norms and software requirements that are represented as phrases.

The process proposed is a concern to define compliance in the software design, in running time, the software adapts to a specification that was already defined primarily. These worksare a concern with compliance between legislation and software requirement and assumes that the software complies with the requirement specification. So, the verification, if the software was built in conformity with the requirement specification, is not the scope of this work.

Another limitation was the validation of this work. To validate the monitor part, we need to monitor laws and requirements for an extended period of the changes happen. Also, the software agents were only specified in this work and not built, so we use an example of how they will behave, but the proposal was not tested in a real case. Although monitor those changes follows the same principle of the context changes, that was shown in the example.

6.3
Future work

The Eunomia Framework was applied only in the context of this research. Even the framework being complex, the application is relevant to the context of requirement engineer, where compliance is growing a lot.

However, the framework needs improvements since it lacks patterns to help the mapping the non-functional requirements from text law to the Desiree template. We conclude that the most challenging part is to understand the legal text, so we need better tool support to extract regulatory requirements from the law. One possible solution can be to search for Non Functional Requirements patterns. Another further direction is improving the automation support for NomosT to Desiree transformation and also for the requirements comparison, we can have a tool that made this comparison, shows the possible conflicts and the possible resolutions.

Also is necessary to think in better ways to mine the text law, one proposal is to do double mining, first searching for rights, obligations, and then searching for the keywords. Another possibility is first discovered the relevant words in the law and then mined using those words. Other possible future work is to use other tools to mine the text law, e.g., the R Software and see the performance compared with NomosT.

Further scenarios are needed to validate the proposal. Future work will be a target for a running example to support this strategy towards software requirements compliance. Also, it is necessary that other requirement engineers be involved as to test the proposal and compare to as ad-hoc situation and with the other methods to verify the effectiveness of the proposal.

Also, it isnecessary to create the patterns in other languages to extend the scope of this work, and the framework can be applied in different countries. One future work is to analyze the Brazilian legislation do discover the patterns of rights, obligations, situations, and mine the “Internet Civil Mark” to discover the regulatory requirements and compare with a software specification that needs to comply with this law.

Another future work is to use the requirement specification generated in this work for the software agent to develop them. With the agent developed, run a case study about environment monitoring, to verify the behaver of the agents in a real situation.

One future work is to extend the Eunomia framework to build the support to legal compliance in configuration management (Leon, 2000) (Hass, 2003) (Conradi e Westfechtel, 1998) (Dart, 1991) (Werner et al., 2010) with the aim of tracking the changes. The idea is to provide support for the system to have continuous compliance despite the changes in requirements, law, and environment where the software is running, verifying where the change was and how affects the system, helping to minimize the impacts of this change. The Configuration Management includes two activities: traceability and control of version. To promote the traceability between requirements and laws helps to calculate the changes

impacts. The control of version will help to deal with the cases of new or updates in laws and requirements. The comparison can occur in different levels: intra-requirements (verify consistency between requirements); intra-laws (verify consistency between laws of different levels –municipal, state, federal or different countries) and between law and requirements crossing the requirements and laws.

Finally, another future research thread is to conduct a more extensive case study in an industrial setting, that includes a more significant number of participants and a complete legal text (not only a few sections). In this way, we will have a more significant set of data to draw conclusions and improve the process.

7
References

ABOWD, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., &Steggles, P. **Towards a better understanding of context and context-awareness**. In Handheld and ubiquitous computing (pp. 304-307). Springer BerlinHeidelberg: September, 1999.

ACM, Sterritt, R. e M. G. Hinchey. **Biologically-inspired concepts for self-management of complexity**. In: 11th IEEE International Conference on Engineering of Complex Computer Systems. 2006. 6.

AGNOLONI, T., Tiscornia, D.: **Extractingnormativecontentfrom legal texts**. In:The 5th MediterraneanConferenceonInformation Systems, MCIS 2010, Tel-AvivYaffoAcademic College, Tel Aviv, Israel: September 12-14, 2010. p. 4 (2010), available in http://aisel.aisnet.org/mcis2010/4

ARMELLIN, G., Chiasera, A., Jureta, I., Siena, A., &Susi, A..**Establishing information system compliance: An argumentation-based framework**. In Research Challenges in Information Science (RCIS), 2011 Fifth International Conference on (pp. 1-9). IEEE (2011).

ANTÓN, A. I., Earp, J. B., Carter, R. A.: **Precluding incongruous behavior by aligning software requirements with security and privacy policies**. In: Information and Software Technology, 45(14), 967-977 (2003).

BASILI, V.; Gianluigi C.; Rombach H. D: **The Goal Question Metric Approach** (PDF). Retrieved 2008-11-12 (1994).

BIAGIOLI, C., Francesconi, E., Passerini, A., Montemagni, S., Soria, C.: **Automaticsemantics extraction in law documents.** In: 10th International Conference on ArtificialIntelligence and Law. pp. 133–140. ICAIL '05, ACM, New York, NY, USA(2005), http://doi.acm.org/10.1145/1165485.1165506

BOELLA, Guido, et al. **Managing legal interpretation in regulatory compliance.** In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law. ACM, 2013.

BREAUX, T. D.; Schaub, F. **Scaling requirements extraction to the crowd:** Experiments with privacy policies. In: Requirements

Engineering Conference (RE), 2014 IEEE 22nd International. IEEE, 2014. p. 163-172.

_____, Vail, M. W. Antón, A. I.: **Towards regulatory compliance: Extracting rights and obligations to align requirements with regulations.** In 14th IEEE International Conference Requirements Engineering (Re'06) ,pp. 49-58. IEEE. (2006).

BRESCIANI, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: **An agent-oriented software development methodology.** Autonomous Agents and Multi-Agent Systems 8(3) (2004) 203–236

BUSINSKA, L., Kirikova, M., Peņicina, L., Bukša, I., Rudzājs, P.: **Enterprise Modeling for Respecting Regulations.** In: PoEM Short Papers 2012 Emerging Topics in the Practice of Enterprise Modelling, Germany, Rostock, pp.106-118. ISSN 1613-0073 (2012).

CARVALHO, S. T., Loques, O., Murta, L. **Dynamic variability management in product lines: An approach based on architectural contracts**. Software Components, Architectures and Reuse (SBCARS), 2010 Fourth Brazilian Symposium on. IEEE, 2010.

CHUNG, L; Nixon, B.; Yu, E; Mylopoulos, J.: **Non-Functional Requirements in Software Engineering.** Massachusetts.USA. Kluwer Academic Publishers. (2000)

CONRADI, R. and Westfechtel, B. **Version Models for Software Configuration Management.**In: ACM Computing Surveys, v.30, n.2, p. 232-282 (1998).

DARDENNE, A., Van Lamsweerde, A., Fickas, S.: **Goal-directed requirements acquisition**. In: Science of computer programming 20(1-2) (1993) 3–50

DART, S., 1991, **Concepts in Configuration Management Systems**, In: International Workshop on Software Configuration Management (SCM), Trondheim, Norway (June), p. 1-18 (1991).

DE SOUZA Cunha, H. (2014). **Desenvolvimento de Software Conscience com Base em Requisitos** (Doctoral thesis, PUC-Rio).

DEY A. K., Salber D., Abowd G. D. **A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, Context-Aware Computing: A Special Triple Issue of Human-Computer Interaction**, Lawrence-Erlbaum March 2002

DOZIER, C., Kondadadi, R., Light, M., Vachher, A., Veeramachaneni, S., Wudali,R.: **Named entity recognition and resolution in legal text.** In: Francesconi, E., Montemagni, S., Peters, W., Tiscornia, D. (eds.) Semantic Processing of LegalTexts: Where the Language of Law Meets the Law of Language. pp. 27–43.Springer Berlin Heidelberg, Berlin, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-12837-0_2

ENGIEL, P., Leite J.C.S.P. and Mylopoulos J (2017) : **A Tool-Supported Compliance Process for Software Systems** In IEEE Eleventh International Conference on Research Challenges in Information Science, Brighton, United Kingdom

FAULK, Stuart R. **Product-line requirements specification (PRS): An approach and case study**.In: Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on. IEEE, 2001.

FERREIRA, M. G., N. Maiden e J. C. S. do Prado Leite. **From Unknown to Known Impacts of Organizational Changes on Socio-technical Systems**.2015. 30 de 11 de 2015. <https://pdfs.semanticscholar.org/3552/9dfa4f9ece179f6434753b1b7735af7c7362.pdf>.

FLOWERDAY, S. & von Solms, R. (2005). **Continuous auditing: verifying information integrity and providing assurances for financial reports.** In: Computer Fraud & Security, 2005(7), 12-16.

GARCÍA-GALAN, J., Pasquale, L., Grispos, G., Nuseibeh, B.: **Towards adaptive compliance**. In: Proceedings of the 11th International Workshop on Software Engineering for Adaptive and Self-Managing Systems,pp. 108-114. ACM (2016).

GHANAVATI, S., Amyot, D., Peyton, L.: **Compliance analysis based on a goal-oriented requirement language evaluation methodology.** In 17th IEEE International Requirements Engineering Conference, RE'09. ,pp. 133-142. IEEE (2009)

_____., Rifaut A., DuboisE., and Amyot D., **Goal oriented compliance with multiple regulations**, In Proc. IEEE 22nd International Requirements Engineering Conference (RE’14), 2014, pp. 73–82.

GOGUEN, Joseph A. **Consciousness studies**. In: Encyclopedia of Science and Religion. MacMillan Reference USA, 2003.

GUARDA P. and Zannone N. **Towards the Development of Privacy-Aware Systems**. In: Information and Software Technology, 2008.

HASS A., **Configuration Management Principles and Practices**, Boston, MA, Pearson Education, Inc, 2003.

INGOLFO, S., & Souza, V. E. S. (2013, May). **Law and adaptivity in requirements engineering.** In: Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2013 ICSE Workshop on (pp. 163-168). IEEE.

_____., Siena, A., Mylopoulos, J., Susi, A., Perini, A.: **Arguing regulatory compliance of software requirements**. Data & Knowledge Engineering 87(0), 279, 29 (2013), http://www.sciencedirect.com/science/article/pii/S0169023X120010 5X

INSFRAN, E., Chastek, G., Donohoe, P., & do Prado Leite, J. C. S. (2014). **Requirements engineering in software product line engineering**. Requirements Engineering, 19(4), 331-332.

KEPHART, J.O., Chess, D.M.: **The vision of autonomic computing.** In: Computer 36(1), 41–50 (2003)

KPMG. (2008). Continuous **Auditing / Continuous Monitoring: Using Technology to Drive Value by Managing Risk and Improving Performance.**

LEITE, JCSP., Yu, Y., Liu, L., Yu, E. S. K., Mylopoulos, J.: **Quality-Based Software Reuse**. CAiSE pp. 535-550 (2005).

LEON, A. A **Guide to Software Configuration Management**, Artech House Publishers (2000).

LI, F., Horkoff J., Borgida A., Guizzardi G., Liu L.,and Mylopoulos J. :**From Stakeholder Requirements to Formal Specifications Through Refinement**. In REFSQ'15, pp 164–180. Springer (2015)

LY,L. T. MaggiF. M., MontaliM., Riderle-MaS. and van der AalstW. M. P., **Compliance monitoring in business processes:**

Functionalities, application, and tool-support, Information Systems, 2015

MASSEY, A. K., Otto, P. N., Hayward, L. J., & Antón, A. I. (2010). **Evaluating existing security and privacy requirements for legal compliance**. In: Requirements engineering, 15(1), 119-137.

_____., Rutledge, R. L., Antón, A. I., Swire, P. P.: **Identifying and classifying ambiguity for regulatory requirements**. In: IEEE 22nd International Requirements Engineering Conference (RE), pp. 83-92. IEEE (2014).

MAXWELL, J. C., Antón, A. I.: **The production rule framework: developing a canonical set of software requirements for compliance with law.** In: proceedings of the 1st ACM International Health Informatics Symposium pp. 629-636. ACM (2010)

MEDEIROS, R. P.; Souza, U. S.; Protti, F.; Murta, L. G. P. Optimal **Variability Selection in Product Line Engineering.** In: International Conference on Software Engineering and Knowledge Engineering (SEKE), 2012, Redwood City. Proceedings..., 2012. p. 635-64 (2012).

MOENS, M., Uyttendaele, C., Uyttendaele, J.: **Information extraction from legal texts:the potential of discourse analysis**. Int. J. Hum.-Comput. Stud. 51(6), 1155–1171(1999), http://dx.doi.org/10.1006/ijhc.1999.0296

MOTIK, Boris, et al. **OWL 2 web ontology language: Structural specification and functional-style syntax.** In: W3C recommendation 27.65 (2009): 159.

NAKAMURA R., et al., **Terminology matching of requirements specification documents and regulations for compliance checking.** RELAW 2015: 10-18

NAMIRIK.,StojanovicN.,**Pattern-baseddesignandvalidationofbusiness processcompliance**,In: On the Move to Meaningful Internet Systems, 2007, pp.59–76

NEKVI, M.R.I., Madhavji, N.H.: **Impediments to regulatory compliance of requirements in contractual systems engineering projects: a case study.** ACM Trans. Manag. Inf. Syst. 5(3),15:1–15:35 (2015)

OLIVEIRA, A. Padua A.; Leite, J. C. S. P.; Cysneiros, L. M.; Cappelli, C.; **Eliciting Multi-Agents Systems Intentionality: From Language Extended Lexicon to i* Models**. In: International Conference of the Chilean Society of Computer Science, 2007, Iquique. Proceedings of the XXVI International Conference of the Chilean Computer Science Society. Los Alamitos: IEEE Computer Society Press, 2007. v. 16. p. 40-49 – ISBN 0-7695-3017-6.

OLIVEIRA, A.; Leite, J.C.S.P; Cysneiros, L.M. **Método ERi*c - Engenharia de Requisitos Intencional**. (2008). WER 2008.

OTTO, P.N. and Antón A. I: **Addressing legal requirements in requirements engineering.** In Proceedings of the IEEE International Requirements Engineering Conference (RE'07). IEEE, Los Alamitos, CA, 5–14.DOI:10.1109/ RE.2007.65 (2007)

ROBALDO, L., Boella, G., Caro, L.D., Violato, A.: **Exploiting networks in law.** In:Proc. oftheNinthInternationalConferenceon Language Resources and Evaluation(LREC-2014), Reykjavik, Iceland, May 26-31, 2014. pp. 1654–1658 (2014), http://www.lrec-conf.org/proceedings/lrec2014/summaries/95.html

SANTOS E.A.P. ,Francisco R. ,VieiraA.D., LouresE.d.F.R., Busetti M. A. : **Modeling business rules for supervisory control of process-aware information systems**, In: Business Process Management Work-shops, 2012, pp.447–458.

SCHMIDT, J. Y., Antón, A. I., Earp, J. B.: **Assessing identification of compliance requirements from privacy policies.** In: Fifth International Workshop on Requirements Engineering and Law (RELAW), pp. 52-61. IEEE. (2012)

SEMMAK, F., Gnaho, C., & Laleau, R. (2008). **Extended Kaos to Support Variability for Goal Oriented Requirements Reuse**. In MoDISE-EUS (pp. 22-33).

SERRANO, M., & do Prado Leite, J. C. S. **Capturing transparency-related requirements patterns through argumentation**. In: 2011 First International Workshop On Requirements Patterns.

SILVA SOUZA, V. E., Lapouchnian, A., Robinson, W. N., & Mylopoulos, J. (2011, May). **Awareness requirements for adaptive systems**. In: Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems (pp. 60-69).

STIEGHAHN, M., & Engel, T. (2009, November). **Law-aware access control: about modeling context and transforming legislation**. In: JSAI International Symposium on Artificial Intelligence (pp. 73-86). Springer Berlin Heidelberg

SUPAKKUL, S., Hill T., Chung L., Tun T.T., Prado Leite J.C.S. :**An NFR pattern approach to dealing with NFRs**, In18th IEEE International Requirements Engineering Conference, 179-188 (2010)

TIAN, X., et al. **A unified cooperative control architecture for UAV missions**. In: International Society for Optics and Photonics, SPIE Defense, Security, and Sensing 2012: 83920X-83920X.

TRAVASSOS G. H., Gurov, D., Amaral, E. A. G.: **Introdução à Engenharia de Software Experimental**, Relatório Técnico ES-590/02, PESC-COPPE (2012).

VAN HILLO, R., & Weigand, H. (2016, June). **Continuous Auditing & Continuous Monitoring: Continuous value?.** In: Research Challenges in Information Science (RCIS), IEEE Tenth International Conference on (pp. 1-11). IEEE (2016).

WERNER, C. M. L.; Murta, L. G. P.; Prudencio, J. G. G.; Fernandes, P. C. C.; Santos, R. P.; Cepeda, R. S. V.; Correa, C. K. F.; Schots, M.; Lyra, F. **Um Processo de Implantação de Gerência de Configuração na Indústria**. In: Workshop de Manutenção de Software Moderna (WMSWM), 2010, Belém. Anais (2010).

WYNER, A., Peters, W.: **On rule extraction from regulations**. In: Atkinson, K. (ed.) JURIX. Frontiers in Artificial IntelligenceandApplications, vol. 235, pp. 113–122.IOS Press (2011), http://dblp.uni-trier.de/db/conf/jurix/jurix2011.html#

_____., Peters, W., Katz, D.: **A case studyon legal case annotation**. In:JURIX (2013)

YU, Eric. **Modelling strategic relationships for process reengineering.** Ph.D. Thesis, University of Toronto (1995)

_____. , Lapouchnian, A., Liaskos, S., Mylopoulos, J., & Leite, J. C. (2008, May). **From goals to high-variability software design.** In: International Symposium on Methodologies for Intelligent Systems (pp. 1-16). Springer Berlin Heidelberg.

YU, Negishi., Shinpei Hayashi., Motoshi Saeki: **Establishing Regulatory Compliance in Goal-Oriented Requirements Analysis.** CBI 2017: 10-18

ZENI, N. ; Seid, E. ; Engiel, P. ; Ingolfo, S. ; Mylopoulos, J.: **Building Large Models of Law with NomosT.** In: The 35th International Conference on Conceptual Modeling (2016).

ZENI, N., Kiyavitskaya, N., Mich, L., Cordy, J., Mylopoulos, J.: **Gaiust: supporting the extraction of rights and obligations for regulatory compliance**. Requirements Engineering pp. 1{22 (2013)

ZDUN, U., Bener, A., & Olalia-Carin, E. L. (2012). **Guest Editors' Introduction: Software Engineering for Compliance.** IEEE Software, 29(3), 24-27.