الجمهورية الجزائرية الديمقراطية الشعبية

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE**

**UNIVERSITÉ Dr. YAHIA FARES DE MEDEA**

**FACULTE DES SCIENCES**

**DEPARTEMENT DE MATHEMATIQUE ET DE L'INFORMATIQUE**



# MEMOIRE DE MASTER

**En vue de l'obtention du Diplôme de**

**Master en Informatique**

**Spécialité: Ingénierie des systèmes et technologie du web**

## THEME

# Spécification ontologique des exigences SysML

**Présenté par**

**M^lle^. MEBREK Soumia** **M^me^. NEDJRAOUI Wissem**

**Devant les jurys**

| | |
|---|---|
| **Mr. RAHIM Messaoud** | **Promoteur** |
| **Mr. KHELDOUN Ahmed** | **Président** |
| **Mr. MAZARI Redha** | **Examinateur** |

***Année Universitaire: 2017/2018***

# *Remerciements*

*Avant tout, nous remercions le Dieu tout puissant de nous avoir donné la force et la volonté pour achever ce modeste travail.*

*Nous tenons à remercier particulièrement nos parents pour leur contribution, leur soutien et leur patience.*

*Nous nous devons remercier le corps professoral et administratif de la Faculté des Sciences, pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.*

*Nous nous adressons tous nos gratitudes et nos remerciement à notre promoteur monsieur RAHIM Messaoud qui s'est toujours montré à l'écoute et qui était très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu nous consacrer.*

*Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.*

*Nous adressons nos plus sincères remerciements à tous les enseignants qui ont suivi notre éducation pendant toute notre vie éducative.*

*Notre dernière remerciement s'adressent à toute personne ayant contribué, de près ou de loin, à la réalisation de ce modeste travail.*

# Dédicace :

*Je dédie ce modeste travail à :*

*À ma tente Hanouna qui a fait de moi ce que je suis aujourd'hui.*

*À mes parents pour leur sacrifices et soutiens tout au long de mes études*

*À mon cher grand-père Aid.*

*À ma chère grand-mère khokha.*

*À ma tentes Hadda et Mariam.*

*À mes chères frères : zino et nassim.*

*À mon chère sœur : Hanan.*

*À Mon oncle Miloud, son épouse wardia, sa belle-fille Mariam,*

*Ses enfants Ishak et Mohamed*

*À Mon oncle Samir, son épouse : Samira, sa belle-fille Hanine,*

*À mon oncle Ahmed et son épouse souhila.*

*À mes adorables sœurs : Samira, khadidja, Fatima et Amina*

*À mes chères amies :*
*Siham, Fouzia, Amina et Yasmina.*
*À* tous ceux qui m'aiment et que j'aime.

***Somia***

# *Dédicace :*

*Je dédie ce modeste travail à tous ceux qui me portent dans leurs coeurs et m'accompagnent dans leurs pensées.*

*A mes chers parents. Que nulle dédicace ne peut exprimer mes sincères sentiments, pour leur patience illimitée, leur encouragement contenu et leur aide, A celui que j'aime beaucoup et qui m'a soutenue tout au long de ce projet : mon mari ABDELMADJID.*

*A mon fils ♥ ABDERRAHMENE ♥*

*A l'âme de mon grand pére MAHFOUDH Allah yerhmo*

*A ma Manina ♥*

*A mes fréres Hssen, Yassine, et ma sœur Amina*

*A mes oncles : Bachir et sa femme Z, Anis et sa femme K, Malek et sa femme X, et mes tantes Ibtissem et Imen*

*A mes beaux-parents.*

*A toute ma famille, et mes très chères amies : Imen , Amina , Soumia*

*A tous les membres de ma promotion.*

*A tous ceux qui ont contribué de près ou de loin pour que ce projet soit possible, je vous*

*dis merci.*

*A tous mes enseignants depuis mes premières années d'études jusqu'à maintenant.*

*A tous ceux qui me sens chers et que j'ai omis de citer.*

***Wissem***

# Résumé :

Les ontologies sont un paradigme puissant pour la représentation et l'échange des connaissances dans le domaine du Web Sémantique qui ne s'intéressent qu'au traitement automatique de la sémantique. Néanmoins, elles doivent être exprimées dans des formalismes qui devraient en tirer pleinement profit, tout en offrant des outils de modélisations simples et efficaces. SysML trouve beaucoup de succès au sein des concepteurs de modèles et a bénéficié, par conséquent, d'un cumul d'expérience très important ainsi qu'une large gamme d'outils dédiés. Cependant, ces outils (Papyrus,..etc) ne permettent pas la définition d'une sémantique formelle ce qui entrave l'inférence automatique sur les modèle (diagramme d'exigence spécialement). Il serait profitable de tirer profit de ces deux puissants outils(les ontologies et SysML) pour rajouter de la sémantique aux modèles conceptuels.Pour arriver à ça, il fallait développer un outil de transformation des exigences SysML en une forme ontologique permettant le raisonnement et l'inférence.



# Mots clés :

Ontologie ; Exigence ; SysML ; Diagramme ; OWLs ; XML ; RDF/RDFS ; Protégé ; sémantique ; JENA ; raisonnement.



# Abstract :

Ontologies are a powerful paradigm for the representation and the exchange of knowledge, in the field of the Semantic Web, that are only interested in automatic semantic processing. However, they must be expressed in a formalism that should take full advantage while providing tools for simple and effective modeling. SysML has been very successful among modelers and has gathered, therefore, a very important experience and a wide range of dedicated tools. However, these tools (Polarsys,..etc) do not allow the definition of a formal semantics which hinders automatic inference on SysML models (especially Requirement Diagram). It would be beneficial to take advantage of these powerful tools (Ontologies and SysML) for adding semantics to the conceptual models. For this, it was necessary to develop a tool for transforming SysML requirements into an ontological form allowing reasoning and inference.

## Key words :

Ontology ; Requirement ; SysML ; Diagram ; OWLs ; XML ; RDF/RDFS ; Protégé ; semantics ; JENA ; raisoning.

# Sommaire

**Introduction générale** **8**

**1 SysML** **10**
1.1 Introduction . . . . . 10
1.2 Ingénierie des systèmes (IS) . . . . . 11
1.3 L'historique de SysML . . . . . 12
1.4 Présentation de SysML . . . . . 12
1.4.1 Les diagrammes de SysML . . . . . 13
1.5 Les outils de SysML . . . . . 24
1.6 Conclusion . . . . . 24

**2 Les ontologies** **25**
2.1 Introduction . . . . . 25
2.2 Le web sémantique . . . . . 25
2.2.1 Les principales composantes du web sémantique . . . . 26
2.3 Les ontologies . . . . . 26
2.3.1 Définition . . . . . 26
2.3.2 Les constituants d'une ontologie . . . . . 27
2.3.3 Les différents types d'ontologies . . . . . 27
2.3.4 Langages de représentation des connaissances . . . . . 28
2.4 Quelques outils pour le web sémantique . . . . . 34
2.4.1 Le projet Annotea . . . . . 34
2.4.2 JENA . . . . . 35
2.4.3 L'Editeur d'Ontologies Protégé . . . . . 36
2.5 Applications des ontologies . . . . . 37
2.5.1 L'utilisation des ontologies dans la sécurité ( détection d'attaques) . . . . . 37

2.5.2 Utilisation du concept de l'ontologie dans le domaine economique . . . 37
2.5.3 Application à la gestion de documents dans le domaine du bâtiment . . . 37
2.5.4 Application au suivi médical et à la coopération autour du dossier d'un patient . . . 38
2.6 Conclusion . . . 38

**3 Conception** **39**
3.1 Introduction . . . 39
3.2 Méthodologie . . . 39
3.3 Conception détaillée . . . 41
3.3.1 Proposition d'une ontologie pour les exigences . . . 41
3.3.2 La transformation des exigences SysML en OWL . . . 48
3.3.3 Analyse sémantique des spécifications SysML . . . 56
3.4 Conclusion . . . 61

**4 Implémentation et étude de cas** **62**
4.1 Introduction . . . 62
4.2 Outil d'analyse SysML2Owl . . . 62
4.2.1 Module de transformation . . . 62
4.2.2 Module sémantique . . . 64
4.3 Etude de cas . . . 65
4.4 Conclusion . . . 75

**Conclusion générale** **76**

# Table des figures


1.1 Le processus de l'ingénierie des systèmes . . . . . 11
1.2 Le positionnement de SysML par rapport à UML . . . . . 13
1.3 les diagrammes de SysML . . . . . 14
1.4 Exemple du diagramme de cas d'utilisation . . . . . 15
1.5 Exemple du diagramme de séquence . . . . . 16
1.6 Exemple du diagramme d'état . . . . . 17
1.7 Exemple du diagramme d'activité . . . . . 17
1.8 Exemple du diagramme de définition des blocs . . . . . 18
1.9 Exemple du diagramme de bloc interne . . . . . 19
1.10 Exemple du diagramme paramétrique . . . . . 20
1.11 Exemple du diagramme de paquetage . . . . . 21
1.12 Exemple du diagramme des exigences . . . . . 23

2.1 Exemple d'un triplet RDF . . . . . 31
2.2 Exemple d'un document RDF/XML . . . . . 31
2.3 Les sous langages de OWL . . . . . 34
2.4 L'editeur d'annotation Annotea . . . . . 35
2.5 Architecture de JENA . . . . . 35

3.1 Les étapes de la démarche proposée . . . . . 41
3.2 Méta modèle du diagramme d'exigence . . . . . 42
3.3 Le graphe de l'ontologie proposée . . . . . 45
3.4 La transformation du modéle des exigences vers le langage OWL 48
3.5 Fusion des parties de l'ontologie . . . . . 56
3.6 Diagramme d'exigence de caméra mobile . . . . . 60

4.1 Démarche générale . . . . . 65
4.2 Diagramme d'exigence du système de détection d'incendie . . 66
4.3 Forme ontologique du système de détection d'incendie . . . . . 67

4.4 La premiére partie du résultat de la transformation du système 68
4.5 La deuxième partie du résultat de la transformation du système 68
4.6 Affichage de toutes les exigences du système . . . . . 69
4.7 Toutes les exigences du système . . . . . 69
4.8 Affichage de toutes les exigences satisfaites par un bloc . . . . 70
4.9 Choix du bloc . . . . . 71
4.10 Toutes les exigences satisfaites par un bloc . . . . . 71
4.11 Affichage de toutes les exigences contenues . . . . . 72
4.12 Choix de l'exigence . . . . . 72
4.13 Les exigences contenues dans l'exigence choisie . . . . . 73
4.14 Affichage de l'impact du changement d'une exigence . . . . . 73
4.15 L'impact du changement d'une exigence . . . . . 74
4.16 Affichage de l'impact du changement d'un bloc . . . . . 74
4.17 Les exigences impactés par le changement du bloc . . . . . 75

# Liste des tableaux


3.1 Détail sur le concept Requirement . . . . . 46
3.2 Détail sur le concept Block . . . . . 46
3.3 Détails sur le concept Attribute . . . . . 47
3.4 Détails sur le concept Stakeholder . . . . . 47
3.5 Détails sur le concept TestCase . . . . . 47
3.6 Détails sur le concept UseCase . . . . . 47
3.7 La transformation d'une exigence . . . . . 49
3.8 La transformation d'un bloc . . . . . 50
3.9 La transformation d'un cas d'utilisation . . . . . 51
3.10 La transformation de la partie prenante . . . . . 51
3.11 La transformation du cas de test . . . . . 52
3.12 La transformation de la relation de raffinement . . . . . 53
3.13 La transformation de la raelation de vérification . . . . . 54
3.14 La transformation de la relation de satisfaction . . . . . 55

# Introduction générale

L'habitude est d'exprimer ce qu'on attend d'un système ou d'un produit sous forme d'un ensemble d'exigences. Une exigence est l'expression d'un besoin qu'un produit ou un service particulier devrait être ou devrait faire. Elle peut être une fonctionnalité, une caractéristique, une limitation qui doit être satisfaite.

L'ingénierie des exigences est un processus clé pour la réussite de tout projet. Elle vise à faire une traduction/représentation des besoins du client et veille à leurs satisfaction durant tout le cycle de vie du système[8].

La possibilité de décrire les exigences d'un système complexe est ajoutée par le langage SysML, qui est un langage de modélisation permettant de décrire tout ou une partie d'un système, d'un point de vue transversal, comportemental ou structurel [24].

Malgré les apports apportés par SysML pour l'ingénierie des systèmes en général, et l'ingénierie des exigences en particulier, ce langage reste limité par le manque d'une sémantique formelle précise, ce qui est considéré un frein en terme de possibilité d'analyse et de validation.

La représentation sémantique des informations a pour but d'ajouter des informations caractéristiques aux informations existantes pour aider à découvrir, relier et inférer des connaissances qui sont vraisemblables. L'ontologie est la base de ce que l'on appelle la représentation sémantique , c'est la définition formelle des concepts relatifs à un domaine et des relations entre ces concepts .

Dans ce contexte, nous proposons de doter le langage SysML d'une sé-

mantique qui permet de raisonner sur les spécifications décrites avec, et ainsi pouvoir analyser et valider les exigences tout au long du cycle de développement. L'idée consiste à décrire par une ontologie les spécifications SysML. Cette ontologie est plus riche sémantiquement à cause de la définition explicite des associations entre les éléments du modèle SysML. Nous utilisons par la suite l'ontologie construite dans la phase d'analyse.

A ce propos, nous proposons un outil permettant à un concepteur/analyste SysML d'analyser ses spécifications. L'analyse est réalisée en proposant des requêtes sémantiques que l'on exécute sur l'ontologie.

Plusieurs types d'analyse peuvent être menés dans ce travail. Il offre la possibilité d'analyse des relations existantes entre les exigences et les différents éléments de modélisation SysML et d'analyse de l'impact du changement.

Ce travail va être organisé en quatre chapitres :

**Chapitre 1 :** Ce chapitre présente le langage SysML et ses différents diagrammes. Particulièrement, nous focalisons sur la présentation du digramme d'exigences.

**Chapitre 2 :** Dans ce deuxième chapitre, nous présenterons le formalisme du web sémantique et des ontologies , ses langages et ses applications .

**Chapitre 3 :** Dans ce chapitre, nous décrirons notre méthodologie pour l'analyse des exigences SysML en spécifiant les différentes étapes pour arriver à la phase d'analyse.

**Chapitre 4 :** Ce dernier chapitre présente l'outil que nous avons développé et illustre notre méthodologie pour l'analyse des exigences SysML par un cas d'étude pratique.

# Chapitre 1

# SysML

## 1.1 Introduction

L'ingénierie système est une approche méthodologique interdisciplinaire permettant de créer (concevoir , développer, faire évoluer et vérifier) des systèmes complexes répondant à un ensemble précis d'exigences commerciales et techniques[23].

La modélisation a pour objectifs d'abstraire la réalité pour mieux comprendre le système à réaliser . L'UML [1] est l'un des langages utilisés pour modéliser, spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel , il fournit des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel [3].

La communauté de l'ingénierie système (IS) a voulu définir un nouveau langage de modélisation pour les ingénieurs système offrant aux ingénieurs système plusieurs améliorations notables par rapport à UML. Ce langage est SysML [2] , c'est un profil d'UML 2.0 fournissant aux ingénieurs un langage de modélisation allant bien au delà des problématiques de l'informatique. Il ajoute la possibilité de représentation des éléments non logiciels et des exigences d'un système.[14]

1. Unified Modeling Language
2. Systems Modeling Language

Ce chapitre présente le langage SysML et chacun des diagrammes proposés par lui, en focalisant sur le diagramme des exigences.

## 1.2 Ingénierie des systèmes (IS)

C'est un processus coopératif et interdisciplinaire de résolution des problèmes, s'appuyant sur les connaissances, méthodes et techniques issues de la science et de l'expérience et reposant sur des approches de modélisation et de validation. Mis en œuvre pour définir, évoluer et vérifier la définition d'un systéme apportant une solution à un besoin opérationnel identifié conformément à des critéres d'éfficacité mesurables qui satisfasse aux attentes et contraintes de l'ensemble de ses parties prenantes[27].
La modélisation a donc été couramment utilisée pour la décomposition fonctionnelle et structurelle du système. Elle permet de maitriser la complexité du système étudié, car chaque modèle donne accés à une représentation abstraite de différents aspects du système [30]. Sa démarche fait appel à trois grands points de vue complémentaires : fonctionnelle, structurelle et comportementale.

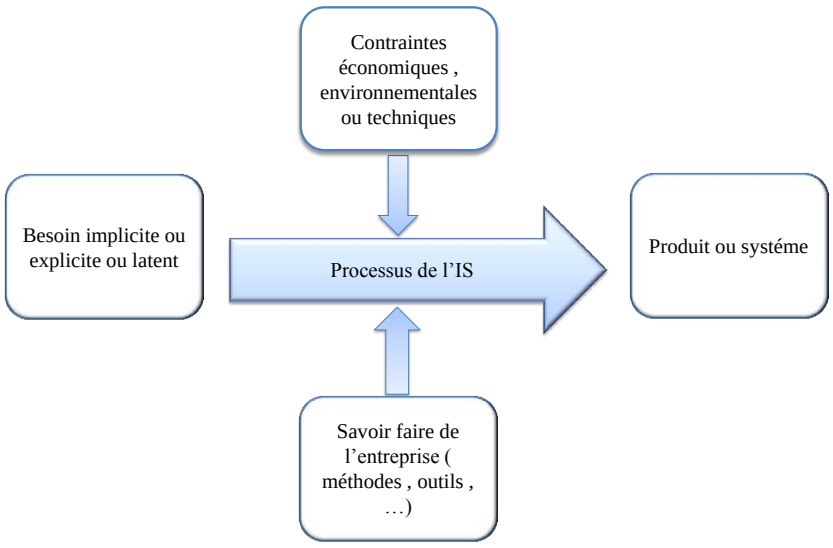



FIGURE 1.1 – Le processus de l'ingénierie des systèmes

## 1.3 L'historique de SysML

Le langage UML permet d'adresser de nombreux besoins pour l'IS, mais il était nécessaire de l'adapter par la définition de « profils UML ». Plusieurs projets ont été menés pour réaliser des profils UML sur des domaines spécifiques de l'IS , comme : le profil MARTE pour les systèmes embarqués en temps réel.

Le besoin de définir un langage basé sur UML pour l'IS a été initié en 2001 par l'organisation internationale de l'ingénierie système INCOSE [3]. Suite à de nombreux ateliers, l'INCOSE et l'OMG [4] ont défini et publié la demande de proposition « UML pour l'IS » en 2003 (UML for SE RFP - Request For Proposal)[26]. Plusieurs membres de l'industrie (BAE, Motorola, oose, Boeing.), éditeurs d'outils (IBM, Sparx Systems.), universités et organisations (INCOSE, AP233) ont travaillés sur la définition du langage de modélisation système SysML (Systems Modeling Language). En juillet 2006 , L'OMG a annoncé l'adoption de SysML , et en Septembre 2007 , les spécification de la première version 1.0 étaient rendues officielles .
La figure 1.2 présente le positionnement de SysML par rapport à UML[17].

## 1.4 Présentation de SysML

SysML est un langage de modélisation spécifique au domaine de l'ingénierie système. Il permet la spécification, l'analyse, la conception, la vérification et la validation de nombreux systèmes et systèmes composés.
SysML est basé sur UML et remplace la modélisation de classes et d'objets par la modélisation de blocs pour un vocabulaire plus adapté à l'Ingénierie Système. Un bloc englobe tout concept logiciel, matériel, données, processus, et même la gestion des personnes[22].
Comme représenté sur le diagramme suivant, SysML réutilise une partie d'UML2 (« UML pour SysML »), et apporte également ses propres définitions (extensions SysML).

3. International Council on Systems Engineering
4. Object Management Group

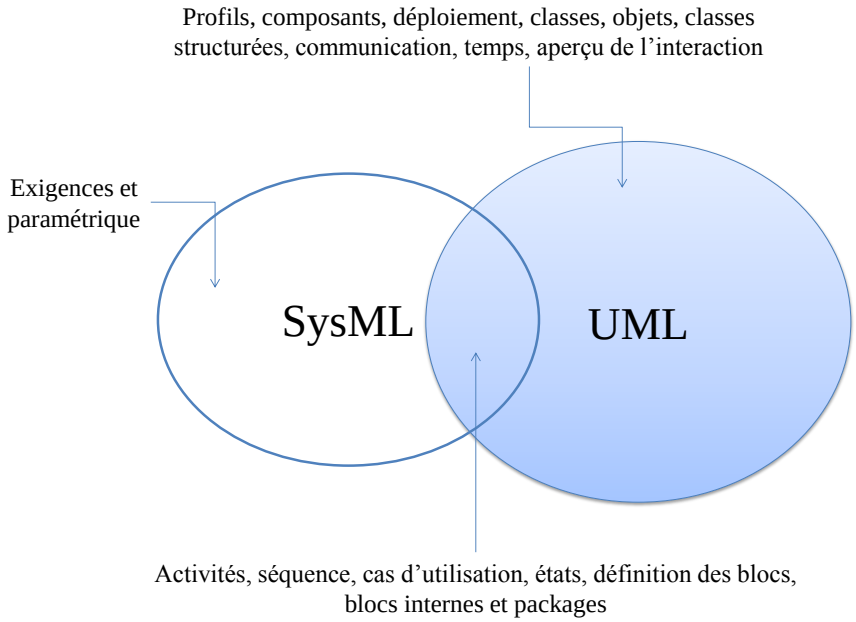



FIGURE 1.2 – Le positionnement de SysML par rapport à UML

### 1.4.1 Les diagrammes de SysML

SysML s'articule autour de neuf types de diagrammes, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système.

On distingue trois types de diagrammes :

— les diagrammes de structure (structure diagram) qui modélisent la constitution du système, ce sont : le diagramme de définition des blocs, le diagramme des blocs internes, le diagramme de paquetage et le diagramme paramétrique.

— les diagrammes de comportement (behavior diagram)qui modélisent le comportement du système, ils sont : Le diagramme de cas d'utilisation , le diagramme de séquence, le diagramme d'activités et le diagramme d'état ;

— le diagramme des exigences (requirement diagram) qui reprend les contraintes du cahier des charges.

La figure 1.3 montre ces diagrammes :

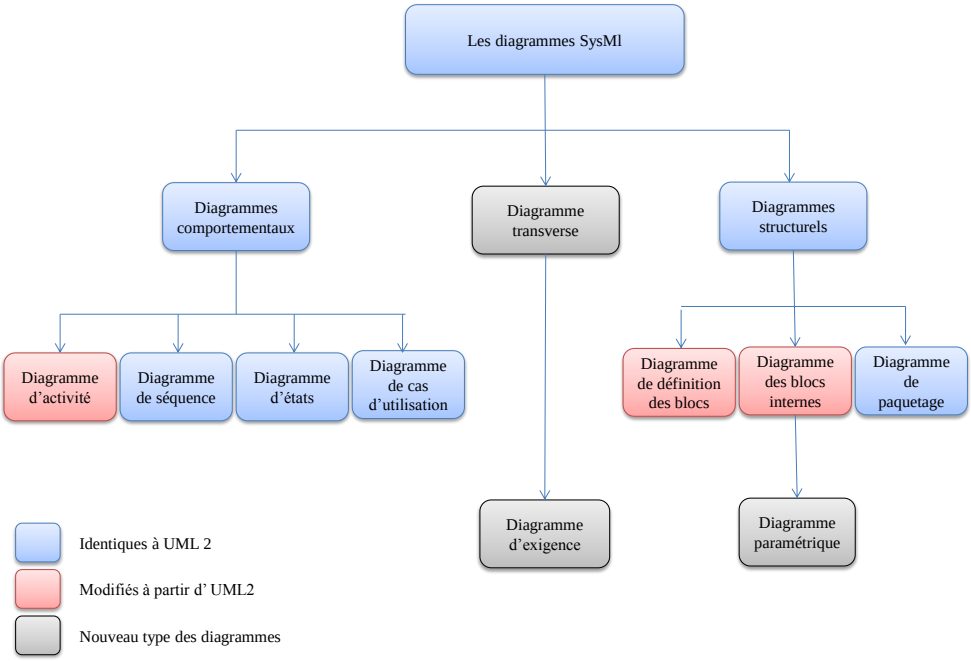



FIGURE 1.3 – les diagrammes de SysML

### Quatre diagrammes comportementaux

1. **Diagramme de cas d'utilisation**
   Le Diagrammes de Cas d'Utilisation peut répondre à la question "A quoi va servir le logiciel ?" IL permet de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système. Il capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Un diagramme de cas d'utilisation est composé de :
   — Les acteurs : ce sont des entités extérieures au système modélisé, et qui interagissent directement avec lui , il peuvent étre : des utilisateurs du systéme ( acteur principal) , des logiciels déjà disponibles à intégrer dans le projet . Un acteur correspond à un role , pas une personne physique
   — Les cas d'utilisation : Ce sont des services rendus à un acteur : des fontionnalités de son point de vue , liées entre eux avec des

relations (inclusion , extension ou généralisation)[2]
La figure ci-dessous représente le diagramme de cas d'utilisation de la balance de HALO :

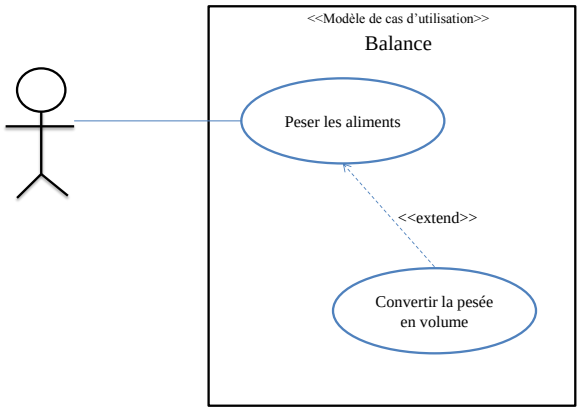



FIGURE 1.4 – Exemple du diagramme de cas d'utilisation

2. **Diagramme de séquence**
   Le diagramme de séquence représente les éléments intervenant dans le scénario, ainsi que les messages échangés dans un ordre chronologique. Les éléments intervenant sont représentés par des lignes de vie. Ces lignes de vies peuvent étre des instances de blocs du modéle. Cette instanciation de blocs établit un lien avec la modélisation structurelle du système. L'ensemble des propriétés du diagramme de séquence utilisées en UML sont également disponibles avec SysML : messages synchrones ou asynchrones, opérateurs(ex : alt, loop, opt, par), références vers d'autres diagrammes de séquence pour ne pas décrire le détail de ces interactions dans le méme diagramme de séquence[3]. La figure 1.5 montre un diagramme de séquence de la balance de HALO :

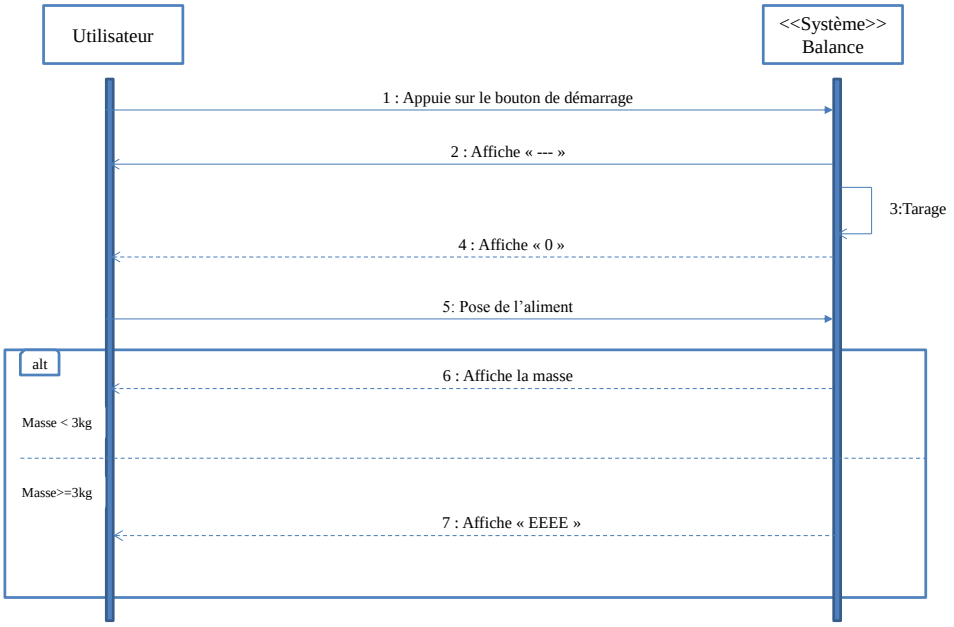



FIGURE 1.5 – Exemple du diagramme de séquence

3. **Diagramme d'état**
   C'est un diagramme dynamique. Il montre les différents états successifs et les transitions possibles des blocs dynamiques. Il représente la succession des états d'un système ou d'un sous-système. Le diagramme d'états est utilisé avec SysML de la méme manière qu'avec UML, c'est- à-dire qu'il permet de représenter le cycle de vie auquel doivent se conformer toutes les instances d'un bloc donné[15].
   La figure 1.6 montre un exemple du diagramme d'état de la balance de HALO :

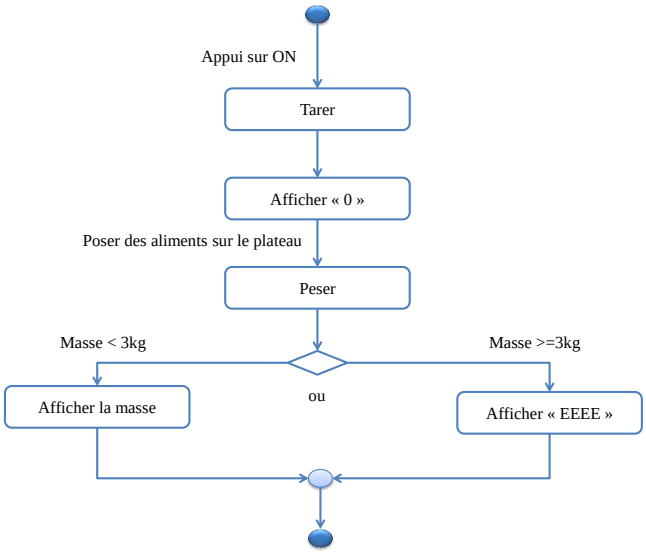



FIGURE 1.6 – Exemple du diagramme d'état

4. **Diagramme d'activité**
   Il permet de représenter le déroulement d'un processus sous la forme d'une activité correspondant à une décomposition séquentielle d'actions. Il est donc particulièrement adapté à la modélisation du cheminement de flots de contrôle et de flots de données [30].

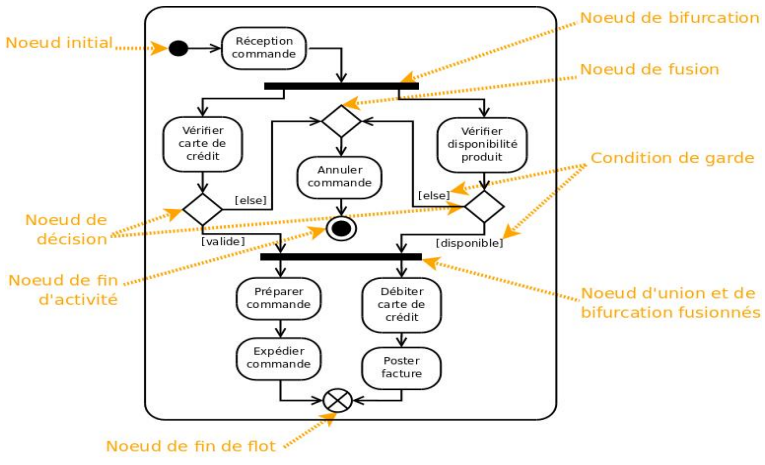



FIGURE 1.7 – Exemple du diagramme d'activité

**Quatre diagrammes structurels**

1. **Diagramme de définition de blocs**
   Il répertorie les constituants du système en précisant éventuellement leur rôle et leur quantité. Chaque bloc peut faire l'objet d'une description plus précise en indiquant ses constituants, ses propriétés, les opérations qu'il peut effectuer ainsi que les contraintes ou limites auxquelles il est soumis.

   Par rapport à UML, le BDD [5] de SysML redéfinit le diagramme de classe en remplaçant les classes par des blocs[5].
   La figure 1.8 montre un exemple de diagramme de définition de blocs (Balance de HALO).

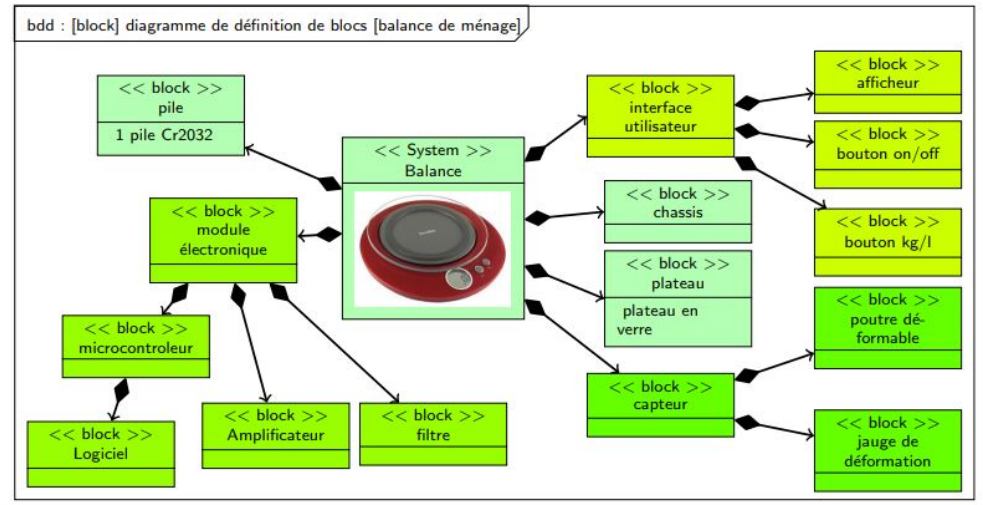



FIGURE 1.8 – Exemple du diagramme de définition des blocs

2. **Diagramme de bloc interne**
   Il représente les liens, les flux et les informations échangées entre les parties d'un bloc ou du système. Le cadre du diagramme représente le bloc lui même ou le système. Donc , il montre l'organisation interne d'un élément statique complexe [30]. La figure 1.9 montre un diagramme de bloc interne de la balance.

5. Block Definition Diagram

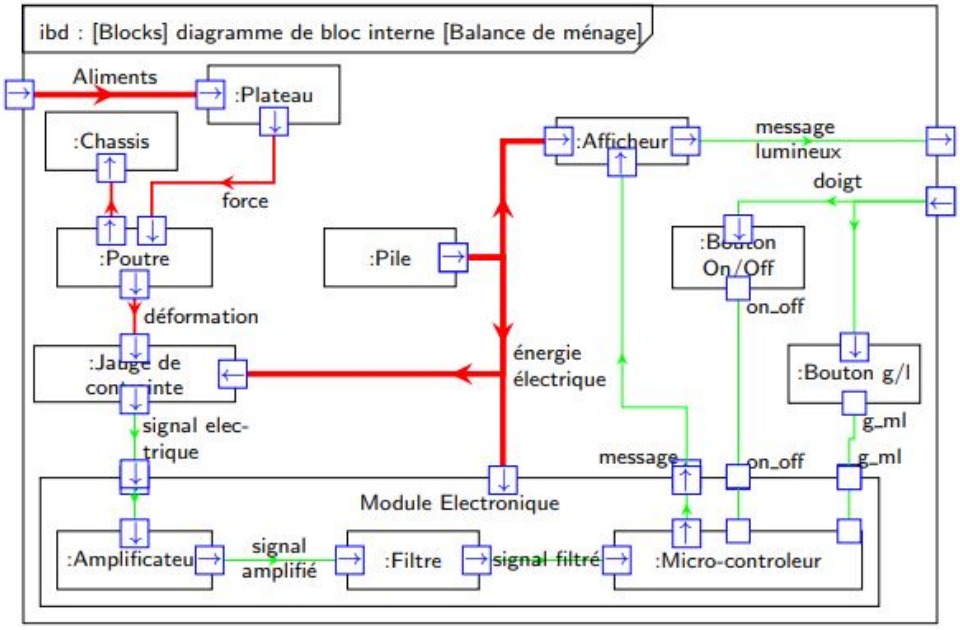



FIGURE 1.9 – Exemple du diagramme de bloc interne

3. **Diagramme paramétrique**
   Le diagramme paramétrique sert à figurer des contraintes sur les valeurs de paramètres système tels que performance, fiabilité, masse, etc.

   Ces contraintes sont définies dans le BDD via des blocs stéréotypés « constraint » et associées au système qui doit les respecter.
   Une contrainte fournit des paramètres accompagnés d'une fonction de transformation (formule mathématique...)[5] .

   La figure 1.10 montre un exemple du diagramme paramétrique :

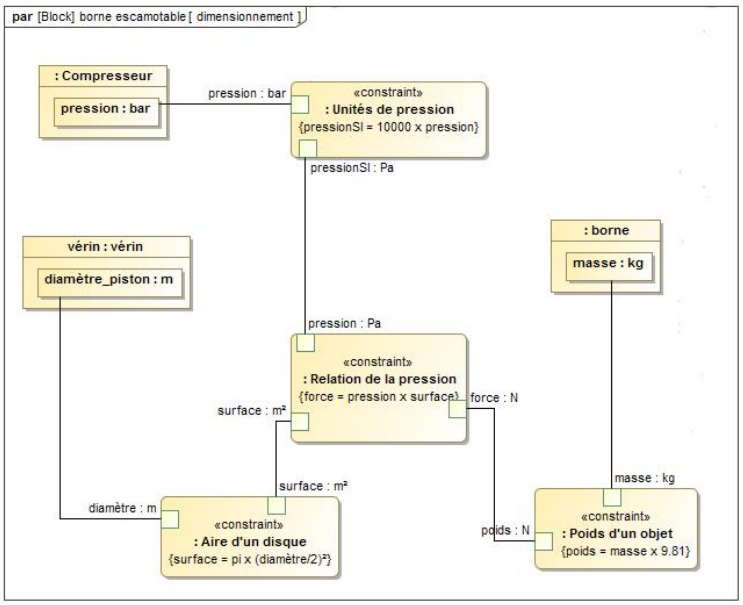



FIGURE 1.10 – Exemple du diagramme paramétrique

4. **Diagramme de paquetage**
   C'est la représentation graphique des relations existantes entre les packages composant un système. Il permet de regrouper sous une méme appellation un ensemble d'éléments de modélisation. Un package est susceptible de contenir n'importe quel élément de modélisation SysML. On l'utilise pour regrouper au sein d'une méme entité, un diagramme de cas d'utilisation, les diagrammes de séquence associés, le diagramme de classes. [13] La figure 1.11 montre un exemple de diagramme de paquetage de la balance :

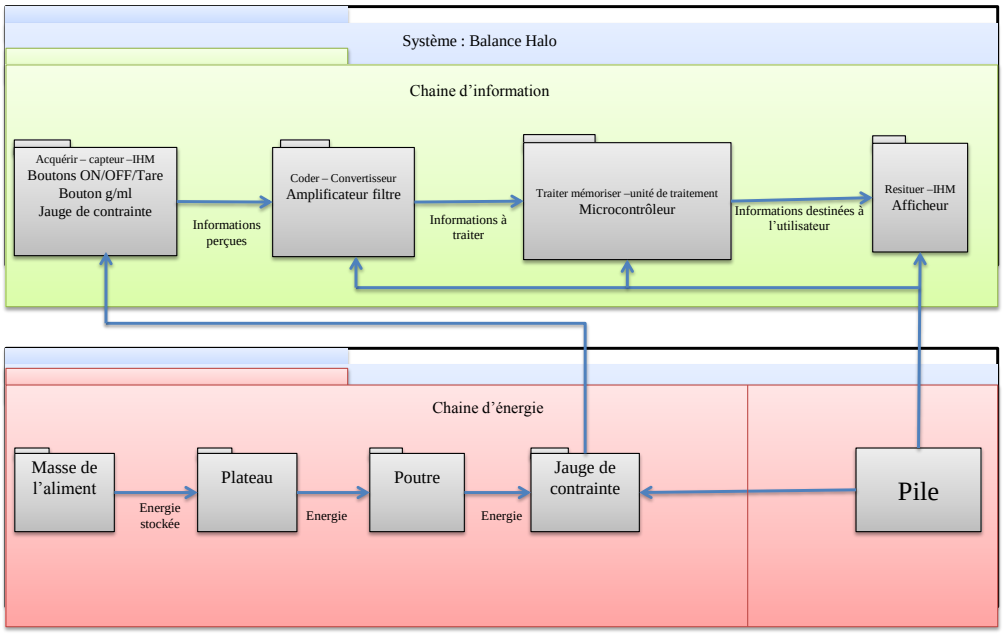



FIGURE 1.11 – Exemple du diagramme de paquetage

### Un diagramme transverse

1. **Diagramme d'exigence**

   Une exigence spécifie un besoin ou une règle qui doit être satisfaite , elle peut spécifier une fonction qu'un système doit exécuter ou des critères de performance à atteindre.

   L'objectif de ce diagramme et de lier les solutions mises en œuvre sur le système avec les besoins définis dans le cahier des charges. Il décrit les exigences de manière graphique, tabulaire, ou arborescente.

   Une exigence est définie comme un stéréotype «requirement» de classe UML soumise à un ensemble de contraintes , elle inclut les propriétés

suivantes : son identifiant unique (id) et sa description textuelle (text) et d'autres propriétés supplémentaires que l'utilisateur peut définir . Plusieurs relations d'exigences sont spécifiées , ce qui permet au modélisateur de relier des exigences à d'autres exigences aussi bien qu'à d'autres éléments de modèles.
On trouve des relations pour définir une hiérarchie d'exigences :

— Des exigences dérivées («deriveReqt») , elles correspondent généralement aux exigences de niveau inférieur de la hiérarchie de système. Par exemple, une exigence d'accélération de véhicule peut impliquer des exigences sur la puissance du moteur, le poids du véhicule et le coefficient de traînée.
— Des exigences satisfaites par d'autres éléments du modèle («satisfy»). Cette relation décrit comment une conception ou un modèle de mise en œuvre satisfait une ou plusieurs exigences.
— Des scénarios de tests vérifiant des exigences («verify») . Dans SysML, un scénario de test ou d'autres éléments de modélisation peuvent être utilisés comme un mécanisme général pour représenter n'importe laquelle des méthodes de vérification ou de simulation. Des sous-classes supplémentaires peuvent être définies par l'utilisateur si nécessaire pour représenter des méthodes de vérification différentes.
— Des exigences affinées («refine»). Par exemple, un cas d'utilisation ou un diagramme d'activité peuvent être utilisés pour affiner une exigence fonctionnelle textuelle.
— Une exigence composite (relation de composition UML) peut contenir des sous-exigences dans une hiérarchie d'exigences. Cette relation permet à une exigence complexe d'être décomposée avec des exigences enfants.
— Des exigences esclaves, ce sont des exigences dont la propriété de description textuelle est une copie (stéréotype «copy») en lecture seule de la même propriété d'une exigence maître liée, cela permet la réutilisation des exigences.
— Une relation d'exigence de trace générique («trace») fournit une relation universelle entre une exigence et un autre élément de modèle. La sémantique de trace laisse de la souplesse au modélisateur car elle n'inclut aucune contrainte réelle. En conséquence, il est recommandé que la relation de trace ne soit pas utilisée en même temps que les autres relations d'exigences décrites ci-dessus.

SysML définit de nouveaux commentaires stéréotypés permettant d'associer une explication à des associations ou éléments du modèle :

— Problem : commentaire dont la description pose le problème ou le besoin qui a donné lieu à la création de l'association ou de l'élément associé

— Rationale : commentaire dont la description indique la raison ou la justification par rapport à l'élément ou l'association associé

Les modélisateurs peuvent personnaliser des taxonomies d'exigences en définissant des sous-classes supplémentaires du stéréotype d'exigence. Par exemple, un modélisateur peut vouloir définir des catégories d'exigences comme des exigences opérationnelles, fonctionnelles, non fonctionnelles, d'interface, de performance, physique, de stockage, d'activation/désactivation, de contraintes de conception et d'autres exigences spécialisées comme la fiabilité et la maintenabilité, ou encore représenter un besoin de partie prenante (sponsor) de haut niveau[5][17]. La figure 1.12 montre un exemple du diagramme d'exigence :

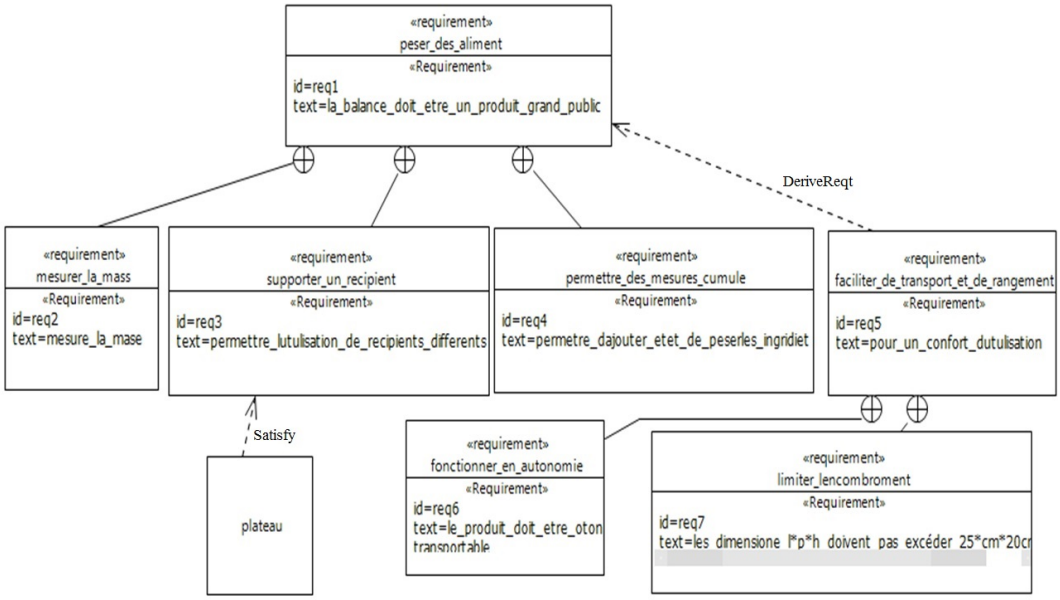



FIGURE 1.12 – Exemple du diagramme des exigences

## 1.5 Les outils de SysML

Le langage SysML a été intégré par de nombreux éditeurs, commerciaux ou open source :

- Sparx Systems Enterprise Architect (plugin SysML ou version Ultimate requise).
- IBM Rational Software Modeler (plugin d'une société tierce disponible).
- Magicdraw (plugin SysML requis).
- Open source : POLARSYS ,Topcased (environnement Eclipse).

  Dans ce travail ,l'environnement POLARSYS est utilisé, qui est une boite à outils open source qui tourne sur un environnement eclipse.[30]

## 1.6 Conclusion

SysML est un profil UML qui réduit le nombre de diagrammes utilisés par rapport à UML tout en y ajoutant des concepts et des nouveaux diagrammes plus adaptés aux approches de modélisation système .

Grace à son expressivité, son pouvoir d'abstraction, sa modularité et son aptitude pour la description de larges spécifications, SysML est un langage suffisamment flexible pour permettre son utilisation dans divers cas de développement, cet aspect positif est constamment mis en valeur. C'est sa grande force, mais en même temps sa plus grande faiblesse, il souffre principalement d'une faible sémantique formelle.

# Chapitre 2

# Les ontologies

## 2.1 Introduction

Les données sur le Web constituent une grande source d'informations représentées sous des formes utiles pour l'être humain mais difficiles pour des traitements automatiques.
Afin de pouvoir les exploiter par des machines et les munir de sémantique , leur contenu doit être décrit en utilisant les méta-données formelles s'appuyant sur des ontologies .
Ces dernières jouent un rôle important dans le Web sémantique, elles sont employées , d'une part , comme une forme de représentation formelle de la connaissance sur un sujet du monde réel et, d'autre part, dans l'objectif de permettre un traitement automatiquement dessus.

## 2.2 Le web sémantique

Le Web sémantique, permet aux machines de comprendre la sémantique, la signification de l'information sur le Web.
Il ne fournit pas des URIs seulement aux documents, comme le fait le Web actuel, mais il le fait aussi pour les concepts et les relations qui définissent et mettent en relation les ressources du Web.
Le Web Sémantique a pour but d'ajouter une information caractéristique au Web pour aider à l'automatisation des services, de découvrir, relier et inférer des connaissances qui sont vraisemblables.
Son objectif principal est de rendre le contenu du web interprétable non

seulement par l'homme , mais aussi par la machine , donc permettre aux utilisateurs une recherche intelligente , en faisant une requête sur le moteur de recherche en langage naturel , il transformera cette demande en langage cohérent et compréhensible par la machine[16].

### 2.2.1 Les principales composantes du web sémantique

L'architecture du web sémantique s'appuie sur une pyramide de langages (architecture en couches) pour représenter des connaissances sur le web en satisfaisant les critères de standardisation, d'interopérabilité et de flexibilité. La liste suivante introduit la fonction principale de chaque couche :

- XML [1] est utilisé comme couche de base syntaxique du web sémantique. Le langage XML est actuellement considéré comme un standard pour le transport de données sur le web.
- La couche RDF [2] représente les métadonnées pour les ressources web.
- La couche « ontologie », fondée sur une formalisation commune, spécifie la sémantique de métadonnées fournies dans le web sémantique.
- La couche « logique » s'appuie sur des règles d'inférence qui permettent le raisonnement intelligent exécuté par des agents logiciels[4].

## 2.3 Les ontologies

### 2.3.1 Définition

Le mot «*ontologie*» vient du mots grec : ontos pour être et logos pour univers et est employé dans des contextes très différents (la philosophie, la linguistique ou l'intelligence artificielle) En informatique elle est définie comme un outil qui permet de structurer un ensemble de concepts afin de leur donner un sens. Elle définit notamment l'ensemble de rapport possible entre ces concepts et permettent de créer des restrictions et des conditions sur ces liens. Une ontologie peut être utilisée pour déduire de nouvelles informations à partir des informations disponibles.

1. EXtensible Markup Language
2. Resource Description Framework

### 2.3.2 Les constituants d'une ontologie

Les connaissances décrivant un domaine spécifié sont représentées par les éléments d'ontologie suivants :

- Concept/classe : Un concept représente un ensemble d'objets et leurs propriétés communes. Les concepts peuvent être une idée, un principe, une notion profonde, ils sont appelées aussi termes ou classes de l'ontologie. Ces concepts peuvent être classifiés en plusieurs niveaux :
  - Niveau d'abstraction (concret ou abstrait).
  - Niveau d'atomicité (élémentaire ou composée).
  - Niveau de réalité (réel ou fictif).
- Individus / Instances : L'individu est une instance de concept, autrement dit, c'est l'élément décrit par le concept, par exemple les individus Wissem et Soumia sont des occurrences du concept Personne.
- Relations : Les relations d'une ontologie désignent différentes interactions et corrélations entre les concepts de l'ontologie. Ces relations réunissent les associations suivants : sous classe de (spécification ou généralisation), partie de (agrégation ou composition), associé a, instance de, est un ... etc.
- Restrictions / règles / axiomes : Pour exprimer toute chose autre que concept ou propriété : Restriction : chose qui doit être vraie pour valider ce qui est exprimé. Règle : affirmation sous la forme : (antécédent >conséquent) décrivant des inférences possibles .
  (ex : oncle(x,y)=>frère(x,z)^père(z,y)). Axiome : assertion générale sur les fondements de l'ontologie, les axiomes sont utilisés pour décrire les assertions de l'ontologie qui seront considérés comme vrais. Cette détermination a pour but de définir les significations des composants d'ontologie, les contraintes sur les valeurs des attributs, et les arguments de relations.
- Fonction : Elles constituent des cas particuliers de relation dans laquelle un élément de la relation, le nième, est défini en fonction des n-1 éléments précédents[28].

### 2.3.3 Les différents types d'ontologies

On peut définir deux classes d'ontologies :

1. une catégorie basée sur la structure de la conceptualisation.
2. une deuxième catégorie basée sur le sujet de la conceptualisation.

Dans la première catégorie, on distingue trois sous catégories :

- les ontologies terminologiques (lexiques, glossaires...),
- les ontologies d'information (schéma d'une BD [3]),
- les ontologies des modèles de connaissances.

Dans la deuxième catégorie, on distingue quatre sous catégories :

- les ontologies d'application : elles contiennent toutes les informations nécessaires pour modéliser les connaissances pour une application particulière.
- les ontologies de domaine : elles fournissent un ensemble de concepts et de relations décrivant les connaissances d'un domaine spécifique.
- les ontologies génériques (dites aussi de haut niveau) : elles sont similaires aux ontologies de domaine, mais les concepts qui y sont définis sont plus génériques et décrivent des connaissances tels que l'état, l'action, l'espace et les composants. Généralement, les concepts d'une ontologie de domaine sont des spécialisations des concepts d'une ontologie de haut niveau.
- Les ontologies de représentation (méta-ontologies) : elles fournissent des primitives de formalisation pour la représentation des connaissances. Elles sont généralement utilisées pour écrire les ontologies de domaine et les ontologies de haut niveau. Exemples : Frame Ontology et RDF Schema Ontology [4].

### 2.3.4 Langages de représentation des connaissances

**XML :** Le langage XML est un langage qui permet de décrire des données à l'aide de balises et de règles que l'on peut personnaliser.
Son objectif est de faciliter les échanges de données entre les machines et décrire les données de manière aussi bien compréhensible par les hommes qui écrivent les documents XML que par les machines qui les exploitent.

XML permet de structurer un document en définissant ses propres balises en fonction des besoins et sans tenir compte ni de la signification de cette structure ni des systèmes informatiques qui vont l'exploiter. Des standards comme XPath et XQuery ont été développés afin de parcourir et d'interroger l'arborescence XML des documents.

3. Base de données

Un document XML est structuré en 3 parties : La première partie, appelée prologue permet d'indiquer la version de la norme XML utilisée pour créer le document. Le prologue se poursuit avec des informations facultatives sur des instructions de traitement à destination d'applications particulières. La syntaxe est la suivante :

< ? Instruction de traitement ?>

La deuxième partie est une déclaration de type du document (à l'aide d'un fichier annexe de type DTD ou schémas XML). Les DTD (Document Type Definition) et les schémas XML permettent de forcer l'utilisation des balises et de leurs imbrications pour la représentation des informations.
De cette façon, XML permet de déclarer, et donc de partager, une syntaxe propre à la représentation d'informations utiles à un domaine ou à une application.
Et enfin, la dernière partie d'un fichier XML est l'arbre des éléments.
Un exemple d'un document XML :

```
<?xml version = "1.0" encoding="UTF-8" standalone="yes" ?>

<boutique>
  <telephone>
    <marque>Samsung</marque>
    <modele>Galaxy S3</modele>
  </telephone>

  <telephone>
    <marque>Apple</marque>
    <modele>iPhone 4</modele>
  </telephone>

  <telephone>
    <marque>Nokia</marque>
    <modele>Lumia 800</modele>
  </telephone>
</boutique>
```

La définition DTD du fichier ci-dessus est la suivante :

```
<!ELEMENT boutique (telephone*)>
<!ELEMENT telephone (marque, modele)>
<!ELEMENT marque (#PCDATA)>
<!ELEMENT modele (#PCDATA)>
```

On distingue deux types de DTD : les internes et les externes.

1. Une DTD interne est une DTD qui est écrite dans le même fichier que le document XML. Elle est généralement spécifique au document XML dans lequel elle est écrite.
2. Une DTD externe est une DTD qui est écrite dans un autre document que le document XML. Si elle est écrite dans un autre document, c'est que souvent, elle est commune à plusieurs documents XML qui l'exploitent[18].

**RDF/RDFS :** RDF (Resource Description Framework) est un modèle de données pour les objets (ressources) et les relations entre eux, fournissant des sémantiques simples pour ce modèle de données qui peuvent être représentés en XML.

RDF est, aussi, un langage pour les métadonnées sur le web. La syntaxe de RDF est basée sur celle de XML. Le modèle de base de ce langage est conçu pour permettre d'associer des attributs aux ressources du Web en utilisant la description de métadonnées sémantiques.

Il est la spécification d'un système d'expression d'assertions sémantiques simples. Ainsi, RDF permet de voir le Web comme un ensemble de ressources reliées par les liens étiquetés «sémantiquement», et permet, alors d'accroître la facilité de traitement automatique des ressources Web.
La structure fondamentale de toute expression en RDF est une collection de triplets, chacun composé d'un sujet, un prédicat et un objet[11].

1. Ressource (Sujet) : une entité d'informations pouvant être référencée par un identificateur. Cet identificateur doit être une URI [4]. Les sujet

4. Universal Resource Identifiers

ou les ressources sont variés : une page Web, une partie d'une page Web, un site Web complet, un objet non accessible par le Web comme un livre...

2. Propriété (prédicat) : un aspect spécifique, un attribut, une caractéristique ou relation utilisée pour décrire une ressource...
3. Valeur : Un littéral (simple chaîne de caractères) ou une ressource.

La figure qui suit montre un exemple de triplet :

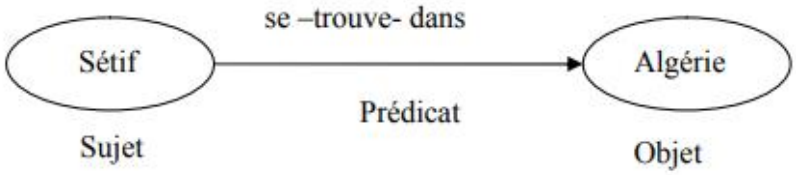



FIGURE 2.1 – Exemple d'un triplet RDF

Un document RDF est un ensemble de triplets . Les éléments de ces triplets peuvent être des URIs, des littéraux ou des variables. Cet ensemble de triplet peut être représenté par un graphe.
Plus exactement, un multigraphe orienté étiqueté où les éléments apparaissant comme sujet où les objet sont des sommets, et chaque triplet est représenté par un arc dont l'origine est son sujet et la destination est son objet.
Ce document sera traduit par un document RDF/XML, et est souvent représenté sous une forme graphique (voir la figure 2.2).

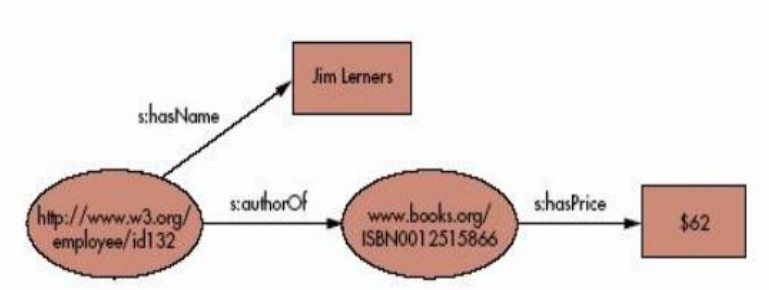



FIGURE 2.2 – Exemple d'un document RDF/XML

RDFS est un méta modèle recommandé par le W3C permettant la définition de schéma/modèle décrivant l'univers sémantique des déclarations RDF. RDFS est le langage de définition de vocabulaires pour RDF, et fournit ainsi un système de typage pour les déclarations RDF.
Il permet la définition des classes et des sous-classes (rdfs :Class, rdfs :subClassOf) décrivant les ressources à annoter et donnant un sens aux propriétés associées aux ressources. Il permet aussi la formulation de contraintes sur les valeurs associées à une propriété afin de lui assurer une signification (rdfs :domain, rdfs :range)[1].

Voici un exemple qui définit une classe « etudiant», et qui instancie de cette classe une ressource <>, et la définit comme sous classe de la classe <> :

```
<rdfs:Class rdf:ID="etudiant">
<rdf:type rdf:resource="http://www.ontoknowledge.org/#DefinedClass"/>
<rdfs:SubClassOf  rdf:resource="#Humain"/>
 </rdfs:Class>
```

**OWL :** OWL (Ontology Web Language) est une extension du RDF pour palier aux insuffisances de ce dernier. En effet, RDF souffre de certaines limites, qui sont, entre autres :

- rdfs :range définit le domaine de valeurs d'une propriété quelle que soit la classe concernée. Par exemple, il ne permet pas d'exprimer que les nourrissons ne boivent que du lait alors que d'autres niveaux d'ages peuvent boire du Soda.
- RDFS ne permet pas d'exprimer que deux classes sont disjointes. Par exemple, les classes des hommes et des femmes sont disjointes
- RDFS ne permet pas de créer des classes par combinaison ensembliste d'autres classes (intersection, union, complément). Par exemple, on veut construire la classe Personne comme l'union disjointe des classes des hommes et des femmes.
- RDFS ne permet pas de définir de restriction sur le nombre d'occurrences de valeurs que peut prendre une propriété. Par exemple, on ne peut pas dire qu'une personne a exactement deux parents.

— RDFS ne permet pas de définir certaines caractéristiques des propriétés : transitivité (par exemple : estPlusGrand-Que), unicité (par exemple : estLePèreDe), propriété inverse (par exemple : mange est la propriété inverse de estMangéPar).

Le langage OWL (Ontology Web Language) devrait être construit sur RDFS tout en disposant d'une syntaxe XML.

OWL permet de définir des classes de manière plus complexe en utilisant les opérateurs ensembliste (et / ou logique) tels que : l'intersection, l'union, la restriction, etc.), des propriétés inverses ou transitives ou bien encore des restrictions de cardinalité sur les propriétés, et cela en se basant sur une logique de description. OWL facilite, aussi, l'interopérabilité au niveau machine du contenu du Web plus que ce qui est déjà supporté par les XML, RDF et RDF Schema en fournissant du vocabulaire supplémentaire avec des sémantiques formelles.

En plus, OWL est doté de trois sous langages offrant des capacités d'expression croissantes et c'est selon les besoins, on choisit le langage qui convient.

— OWL Lite est le sous langage de OWL le plus simple. Il est destiné aux utilisateurs qui ont besoin d'une hiérarchie de concepts simple. OWL Lite est adapté, par exemple, aux migrations rapides depuis d'anciens thésaurus.
— OWL DL est plus complexe que OWL Lite, permettant une expressivité bien plus importante. OWL DL est fondé sur la logique descriptive (d'où son nom, OWL Description Logics), un domaine de recherche étudiant la logique, et conférant donc à OWL DL son adaptation au raisonnement automatisé.
  Malgré sa complexité relative face à OWL Lite, OWL-DL garantit la complétude des raisonnements (toutes les inférences sont calculables) et leur décidabilité (leur calcul se fait en une durée finie).
— OWL Full est la version la plus complexe d' OWL, mais également celle qui permet le plus haut niveau d'expressivité.
  OWL Full est destiné aux situations où il est plus important d'avoir un haut niveau de capacité de description, quitte à ne pas pouvoir garantir la complétude et la décidabilité des calculs liés à l'ontologie. OWL Full offre cependant des mécanismes intéressants, comme par exemple la possibilité d'étendre le vocabulaire par défaut de OWL[19].

La figure 2.3 montre les sous langages de OWL :

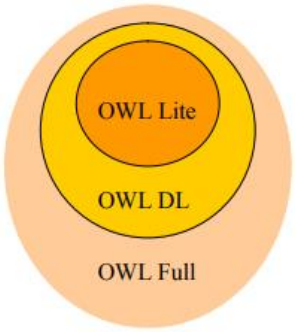



FIGURE 2.3 – Les sous langages de OWL

Il existe entre ces trois sous langage une dépendance de nature hiérarchique : toute ontologie OWL Lite valide est également une ontologie OWL DL valide, et toute ontologie OWL DL valide est également une ontologie OWL Full valide[21][4].

## 2.4 Quelques outils pour le web sémantique

### 2.4.1 Le projet Annotea

C'est un projet du W3C pour faire des annotations web, elles sont des commentaires, des notes ou d'autres types de remarques externes qui peuvent être associésau document web. Une fois qu'on a un document on peut charger les annotations le concernant depuis un serveur afin de les exploiter. On utilise un schéma basé sur RDF pour décrire les annotations comme des métadonnées. Xpointer est utilisé pour localiser des annotations dans un document annoté. Les annotations sont stockées dans des serveurs qui peuvent être interrogés[12].

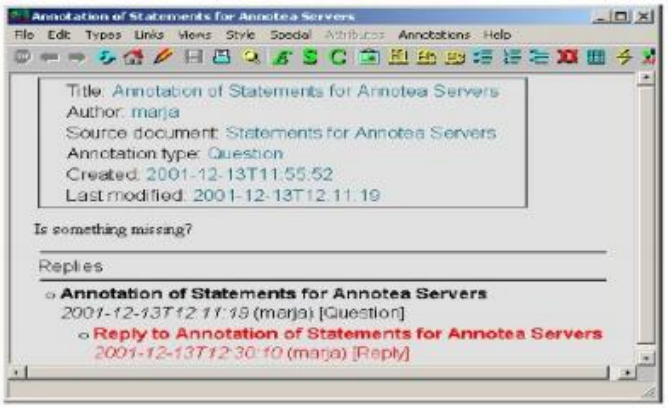



FIGURE 2.4 – L'editeur d'annotation Annotea

### 2.4.2 JENA

Jena est un framework en Java qui permet de construire des applications Web Sémantique. Il fournit un environnement de programmation pour RDF, RDFS et OWL. Il inclut également un moteur de règles d'inférence. Le framework contient :

- Un api pour la lecture et l'écriture de RDF en RDF/XML, N3, et N-Triples.
- Un api OWL.
- Un système de persistance.
- RDQL, un langage de requête pour RDF.

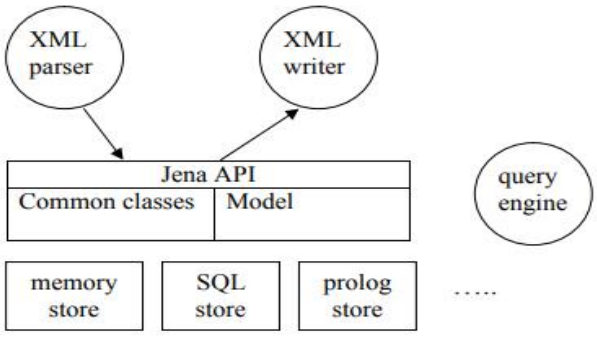



FIGURE 2.5 – Architecture de JENA

### 2.4.3 L'Editeur d'Ontologies Protégé

Protégé est un éditeur qui permet de construire des ontologies pour un domaine donné, de définir des formulaires d'entrée de données, et d'acquérir des données à l'aide de ces formulaires sous forme d'instances de cette ontologie.

Protégé est aussi une plate-forme extensible, grâce au système de plug-ins, qui permet de gérer des contenus multimédias, interroger, évaluer et fusionner des ontologies, etc.

L'outil Protégé possède une interface utilisateur graphique (GUI) lui permettant de manipuler aisément tous les éléments d'une ontologie : classe, propriété, instance,... etc. Il peut être utilisé dans n'importe quel domaine où les concepts peuvent être modélisés en une hiérarchie des classes.

Protégé permet aussi de créer ou d'importer des ontologies écrites dans les différents langages d'ontologies tel que : RDF-Schéma, OWL, DAML, OIL, ... etc. Cela est rendu possible grâce à l'utilisation de plugins qui sont disponibles en téléchargement pour la plupart de ces langages.

Il a deux outils distincts :

— Protégé-Frame permet de créer facilement une interface graphique afin de gérer une ontologie. Cet outil ne demande aucune notion de programmation. Il génère automatiquement les formulaires nécessaires en se basant sur le schéma d'ontologie créé. Il offre également la possibilité de personnaliser l'interface selon les besoins de l'utilisateur

— Protégé-OWL est une extension de Protégé qui supporte le langage OWL. Il permet de décrire plus précisément les classes, les propriétés et les instances grâce aux nombreuses propriétés offertes par OWL. Il est également possible d'interroger un raisonneur via une interface DIG afin de contrôler l'intégrité du modèle et de créer un modèle d'inférences[10].

## 2.5 Applications des ontologies

### 2.5.1 L'utilisation des ontologies dans la sécurité ( détection d'attaques)

Dans l'article « Ontology for Attack detection : An intelligent Approach to web application Security [ABDUL RAZZAQ et AL : Avril 2014 ] les auteurs ont utilize les ontologies pour détecter des attaques web :

- Modéliser les protocoles de communication web
- Prendre en compte les différentes technologies utilisées par les pirates
- Prendre en compte l'impact sur les composantes du systéme
- Concentrer sur les parties où le script malveillant est possible [6]

### 2.5.2 Utilisation du concept de l'ontologie dans le domaine economique

Dans l'article « Ontology-based economics knowledge sharing System [Donghee Yoo et al , 2013] « les auteurs ont utilize le concept des ontologies pour la création d'un systéme de partage des connaissances du domaine economique :

- Modéliser les documents et les connaissances economiques par une ontologie .
- Développement d'une application web pour l'enregistrement , partage et navigation dans ces connaissances .

L'objectif est de montrer la nécessité du partage de connaissances de l'économie et de montrer qu'elle peut étre utilisée avec les technologies du web sémantique[7].

### 2.5.3 Application à la gestion de documents dans le domaine du bâtiment

Le logiciel Aprobatiom permet de guider la recherche de documents lors de la conception d'un nouveau bâtiment, afin d'identifier et de réutiliser des documents traitant de la réalisation de bâtiments ressemblant au nouveau projet. L'ontologie est utilisée pour guider l'utilisateur dans l'expression d'une requête (par exemple, en montrant les contraintes des signatures des relations, c'est-à-dire les instances qui peuvent être reliées). Elle sert aussi pour

caractériser et comparer les différents documents gérés (par exemple, pour vérifier les types en prenant en compte la subsomption, c'est-à-dire les liens hiérarchiques entre catégories)[9].

### 2.5.4 Application au suivi médical et à la coopération autour du dossier d'un patient

Une décision thérapeutique peut impliquer plusieurs experts médicaux et peut aussi nécessiter une connaissance détaillée des antécédents du patient. Le système Ligne de vie repose sur une ontologie pour décrire les symptômes, les diagnostics, les options et les choix thérapeutiques, intégrer différentes contributions au dossier d'un patient et permettre une collaboration non ambiguë des personnes et des systèmes. L'ontologie est utilisée ici comme référentiel commun dans une activité collaborative[9].

## 2.6 Conclusion

Dans ce chapitre nous avons présenté le Web sémantique, qui permet l'exploitation des ressources sur le web par les humains et les machines. Les connaissances sont représentés dans lui sous forme des ontologies .Ces dernières incluent généralement une organisation hiérarchique des concepts pertinents et des relations qui existent entre eux, ainsi que des règles et axiomes qui les contraignent. L'introduction d'une ontologie dans un système d'information vise à réduire, éliminer, la confusion conceptuelle et terminologique et à tendre vers une compréhension partagée pour améliorer la communication, le partage, l'interopérabilité et le degré de réutilisation possible.

# Chapitre 3

# Conception

## 3.1 Introduction

La représentation ontologique des exigences SysML vise à donner de la sémantique au modèle d'exigences SysML afin d'améliorer l'exhaustivité, identifier les données manquantes, incomplètes et incohérentes, améliorer la qualité des connaissances et pouvoir raisonner sur les exigences capturées en utilisant les techniques des ontologies. Cela aide à éliminer un tas de problèmes dans le domaine de l'ingénierie des systèmes.

Le présent chapitre décrit notre démarche pour la représentation ontologique des exigences SysML. Nous commençons par donner une description globale de la démarche proposée.
Puis nous passons à la conception détaillée en présentant notre ontologie proposée pour décrire des exigences et les règles de transformation des exigences SysML en langage OWL.

Enfin, nous discutons l'utilité de la représentation ontologique proposée en proposant un ensemble de requêtes sémantiques concernant le diagramme d'exigences.

## 3.2 Méthodologie

Malgré la richesse du SysML , il reste un langage qui manque d'une sémantique précise .

Notre démarche a pour objectif de doter SysML de mécanismes qui permettent de pouvoir raisonner sur les spécifications SysML en proposant une représentation ontologique des exigences SysML .

Elle permet de laisser le concepteur utiliser le langage SysML pour la phase de spécification mais aussi lui fournir un outil de Question/Réponse pour lui permettre d'analyser ces spécifications en se basant sur la représentation ontologique proposée .

Pour atteindre ces objectifs , notre démarche commence tout d'abord par la définition d'une ontologie écrite en langage OWL . nous avons proposé cette ontologie en se basant sur le méta-modèle du langage SysML tout en pensant à la phase d'analyse (Question/Réponse) . La représentation ontologique des exigences SysML permet de donner de la sémantique au diagramme d'exigence .

La deuxième étape de notre approche consiste à transformer d'une façon automatique les diagrammes d'exigence en langage OWL . Il s'agit de définir l'instanciation des diagrammes d'exigence par rapport aux concepts de l'ontologie proposée.

La troisiéme étape consiste à regrouper l'ontologie ( partie concepts avec les individus générés à partir du diagramme d'exigence ) .

Et la derniére consiste à donner la possibilité aux analystes SysML de raisonner sur les spécifications et analyser les exigences et leurs relations avec les autres éléments de modélisation . Ceci est réalisé à travers le développement d'un framework permettant d'exécuter des requêtes sémantiques (SPARQL) sur les spécifications SysML tout en restant proche du langage SysML (cacher la complexité du langage SPARQL aux analystes SysML ).

La figure 3.1 montre les étapes de notre démarche :

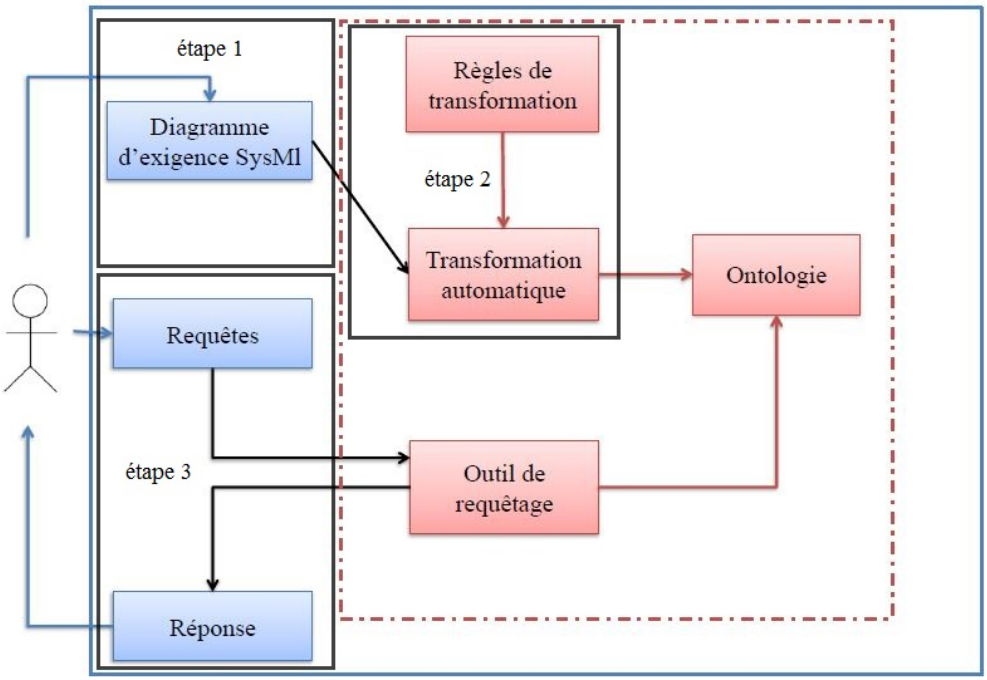



FIGURE 3.1 – Les étapes de la démarche proposée

## 3.3 Conception détaillée

### 3.3.1 Proposition d'une ontologie pour les exigences

Afin de pouvoir représenter les exigences SysML en langage OWL , nous avons procédé à la définition de l'ontologie suivante en se basant sur le méta modèle du langage SysML présenté dans la figure 3.2 [20] :

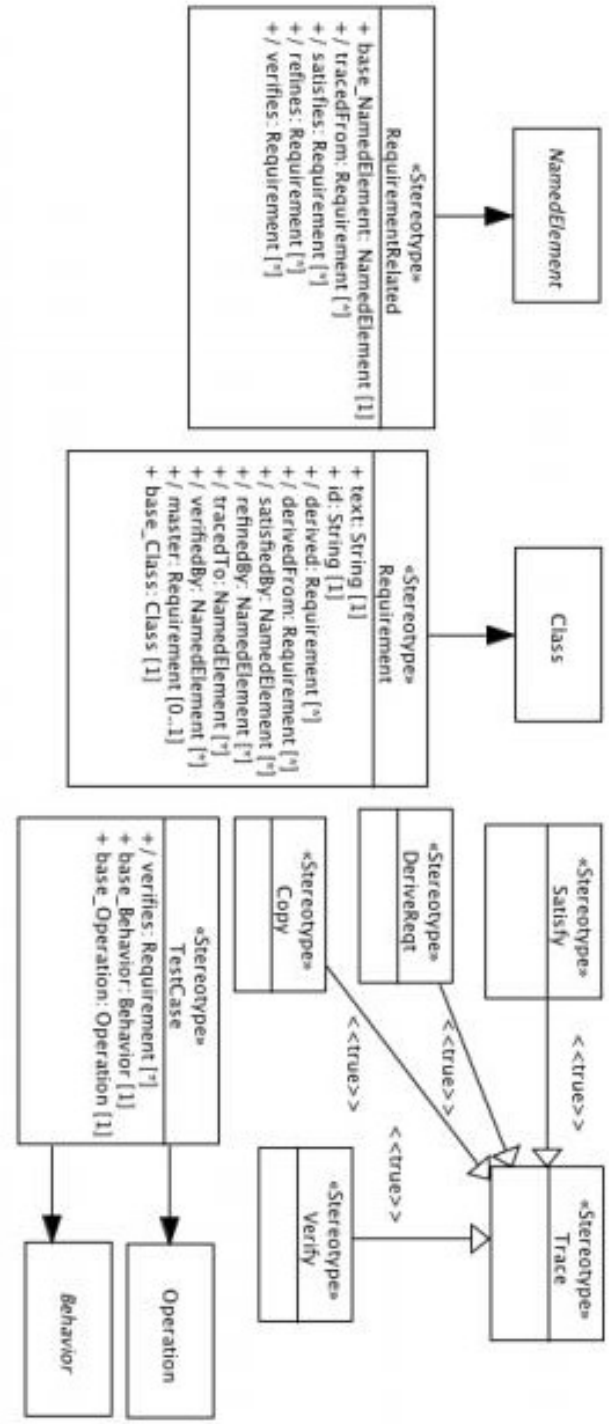





FIGURE 3.2 – Méta modèle du diagramme d'exigence

**Définition des concepts :**

1. **Une exigence (Requirement ) :**
   Une exigence a un identifiant et un texte de description , et un des quatre types :
   - Exigence fonctionnelle : Une exigence concernant un résultat de comportement qui doit être fournit par une fonction d'un système , s'un composant ou d'un service.
   - Exigence non-fonctionnelle : ou exigence de qualité , c'est une exigence relative à un problème de qualité d'une exigence fonctionnelle.
   - Exigence système : une exigence qui concerne la plateforme du logiciel , par exemple : la taille de RAM , architecture matérielle .. etc
   - Exigence de processus : une exigence qui concerne les normes de développement , les directives et les méthodes qui doivent être suivies.
2. **Un bloc (Block) :**
   Un élément matériel qui fait partie d'un autre diagramme et qui satisfait une ou plus des exigences .
3. **Un cas de test (Test Case) :**
   C'est une vérification de l'exigence , elle peut être faite par : un diagramme de séquence , un diagramme d'activité ou un diagramme d'état .
4. **Les attributs :**
   Ce sont des propriétés qu'une exigence peut avoir :
   — Niveau de risque : le risque est un événement qui menace le succès de la réalisation d'un produit ( développement d'un système ) , il peut être : élevé , moyen , bas .
   — Etat : valide ou non .
   — Méthode de vérification : par un teste ou par un modèle- checking (c'est une technique de vérification automatique de systèmes informatiques (logiciels, circuits logiques, protocoles de communication). Il s'agit de tester algorithmiquement si un modèle donné du système lui-même, ou une abstraction du système, en satisfait une spécification logique et une propriété, généralement formulée en termes de logique temporelle.).

5. **Le cas d'utilisation (UseCase)** :
   C'est une description des interactions possibles entre les acteurs et le système .
6. **La partie prenante (Stakeholder ) :**
   C'est une personne ou organisation qui a une influence (directe ou indirecte) sue les exigences d'un système , elle agit comme auteur principale [25] .

**Définition des relations entre les concepts :** L'exigence peut avoir des relations avec tous les autres concepts ( comme nous avons dit dans le premier chapitre ), elle peut :

- Etre dérivée d'une autre exigence .
- Etre copiée d'une autre exigence .
- Faire partie d'une autre exigence .
- Etre tracée par une autre exigence .
- Etre Raffinée par un cas d'utilisation ou par une exigence .
- Etre Satisfaite par un bloc .
- Etre Vérifiée par un cas de teste .
- Etre Reliée à une partie prenante (Stakeholder) .
- Avoir des attributs : état , niveau de risque et méthode de vérification .
- Avoir un cas d'utilisation .

Le graphe de la figure 3.3 montre l'ontologie proposée :

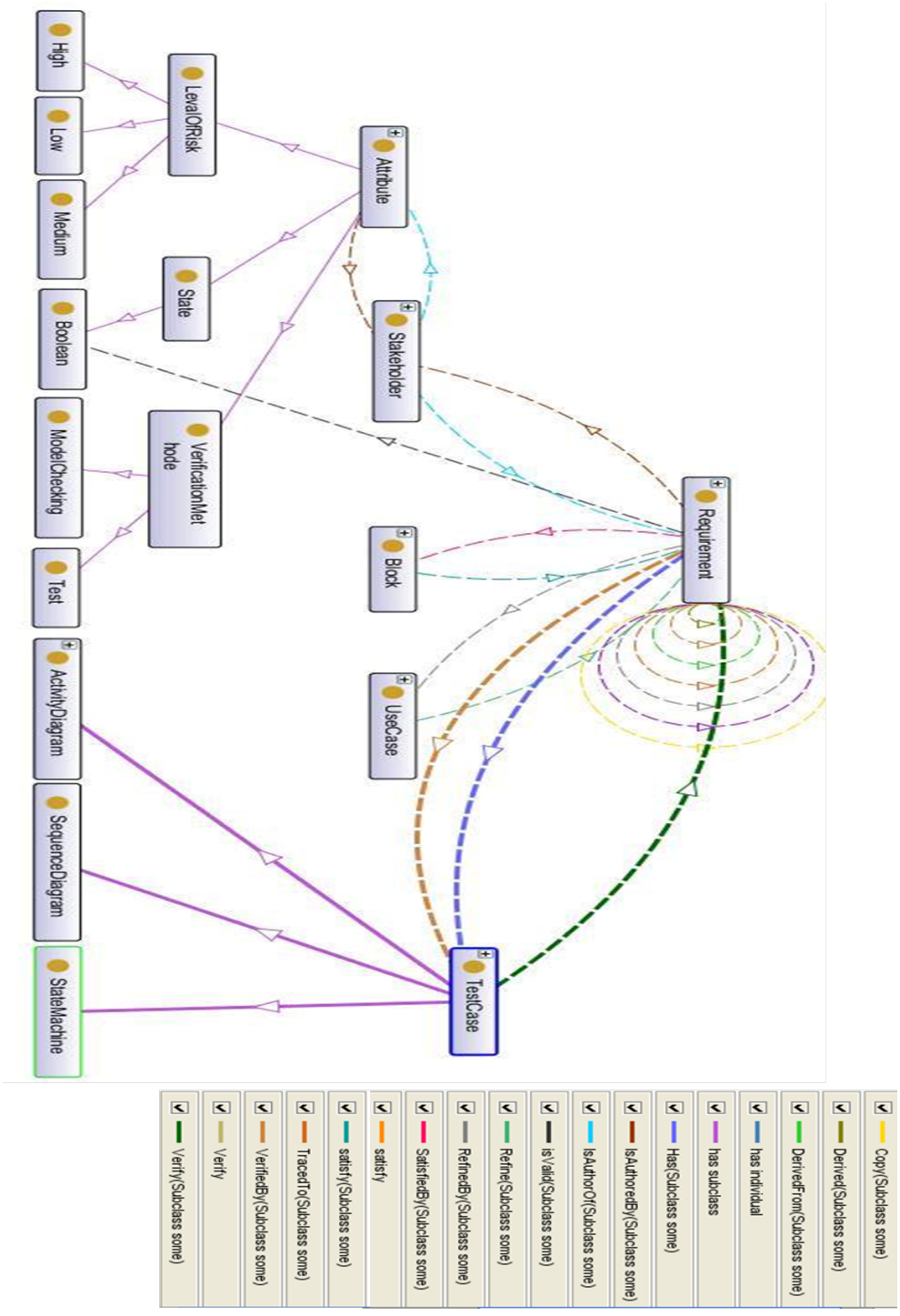



FIGURE 3.3 – Le graphe de l'ontologie proposée

**Détails sur les concepts :**

- Concept exigence (Requirement) :

| Concept | Exigence (Requirement) |
|---|---|
| Sous classes | Exigence fonctionnelle , exigence<br>non-fonctionnelle , exigence du systéme , exigence de processus |
| Restrictions | RefinedBy some requirement<br>isValid some Boolean<br>Contain some requirement<br>Copy some requirement<br>Derived some requirement<br>DerivedFrom some requirement<br>Id only anyURI<br>IsAuthoredBy some Stakeholder<br>RefinedBy some UseCase<br>SatisfiedBy some Block<br>Text exactly 1 anyURI<br>TracedTo some Requirement<br>VerifiedBy some TestCase<br>VerifiedBy some Requirement<br>Has some TestCase<br>Has some Attribute |

TABLE 3.1 – Détail sur le concept Requirement

- Concept bloc (Block) :

| Concept | Bloc(Block) |
|---|---|
| Sous classes | / |
| Restrictions | Satisfy some Requirement |

TABLE 3.2 – Détail sur le concept Block

- Concept Attribut (attribute ) :

| Concept | Attribut (Attribute) |
|---|---|
| Sous classes | Niveau de risque (LevelOfRisk) , état (State) , Méthode de vérification (VerificationMethod) |
| Restrictions | IsAuthoredBy some Stakeholder |

TABLE 3.3 – Détails sur le concept Attribute

- Concept Partie prenante (Stakeholder) :

| Concept | Partie prenante (Stakeholder) |
|---|---|
| Sous classes | / |
| Restrictions | IsAuthorOf some Requirement<br>IsAuthorOf some Attribute |

TABLE 3.4 – Détails sur le concept Stakeholder

- Concept Cas de test (TestCase ) :

| Concept | Cas de test (TestCase) |
|---|---|
| Sous classes | / |
| Restrictions | Verify some Requirement |

TABLE 3.5 – Détails sur le concept TestCase

- Concept cas d'utilisation(UseCase) :

| Concept | Cas d'utilisation (UseCase) |
|---|---|
| Sous classes | / |
| Restrictions | Refine some Requirement |

TABLE 3.6 – Détails sur le concept UseCase

### 3.3.2 La transformation des exigences SysML en OWL

Afin de pouvoir retirer des connaissances concernant la validité ou la qualité des exigences dans un diagramme de SysML , il fallait transformer ce diagramme en sa forme ontologique (écrite en OWL) pour lui mener de la sémantique donc pouvoir raisonner sur lui, cette transformation est de type transformation du modèle à un texte.

La transformation est réalisée en appliquant des régles pour chacun des concepts et des relations.

La figure 3.4 montre comment elle se fait

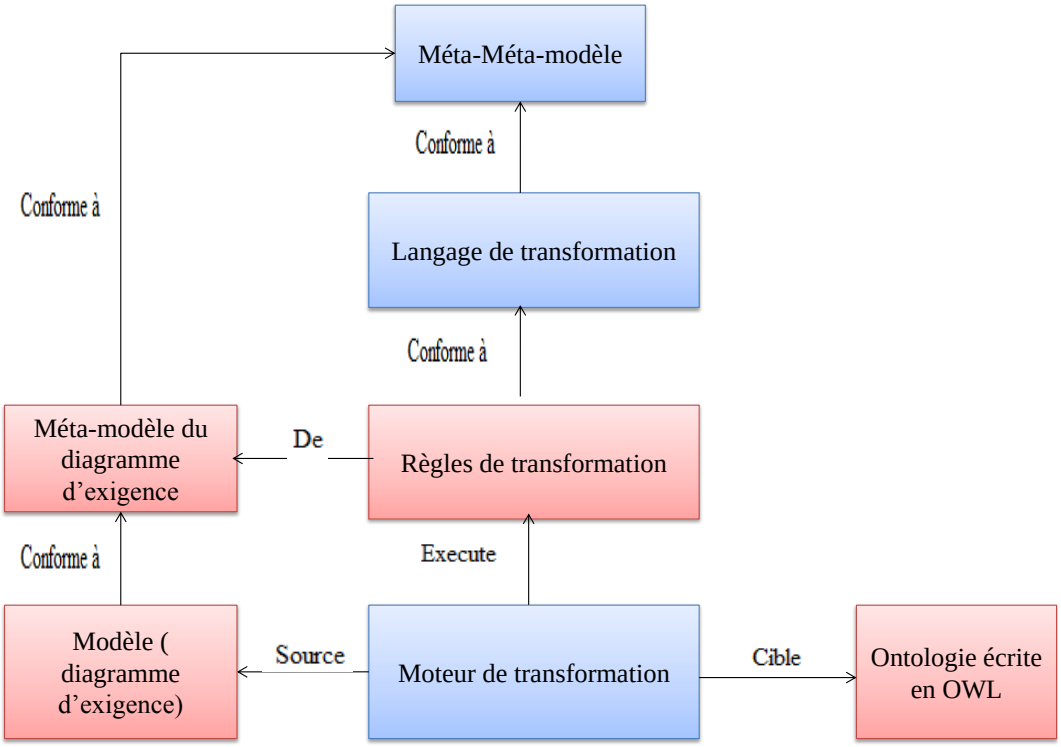



FIGURE 3.4 – La transformation du modéle des exigences vers le langage OWL

Les tableaux suivants montrent comment la transformation (Modéle to Text) de chaque concept SysML vers un concept ontologique est faite :

- **Transformation d'exigence :**
  Chaque exigence est transformé de sa forme SysML vers une forme ontologique.

  Dans le tableau suivant ,on prend l'exemple de la transformation de l'exigence req1 ayant l'identifiant : 001 et un texte descriptif , son type est "exigence fonctionnelle" :

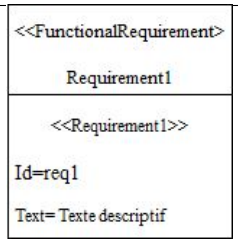
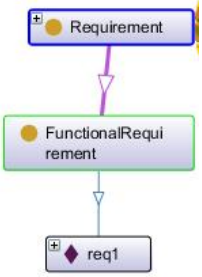
| Concept | Requirement |
|---|---|
| Forme SysML |  |
| Forme ontologique |  |
| RDF/XML | <owl :NamedIndividual rdf :about="www.ex.com/req#req1"><br><rdf :type<br>rdf :resource="www.ex.com/req#FunctionalRequirement"/><br><Text>Texte descriptif</Text><br></owl :NamedIndividual> |

TABLE 3.7 – La transformation d'une exigence

- **Transformation du bloc :**
  L'élément Bloc est transformé de sa forme SysML en une forme ontologique , le tableau suivant montre un exemple de transformation du bloc B1 :

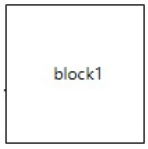
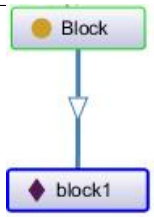
| Concept | **Block** |
|---|---|
| Forme SysML |   |
| Forme ontologique |   |
| RDF/XML | <owl :NamedIndividual<br>rdf :about="www.ex.com/req#block1"><br><rdf :type rdf :resource="www.ex.com/req#Block"/><br></owl :NamedIndividual> |

TABLE 3.8 – La transformation d'un bloc

- **Transformation du cas d'utilisation :**
  Un cas d'utilisation est représenté en SysML par un cercle , il est transformé à un concept ontologique comme le montre le tableau suivant :

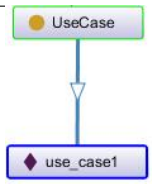
| Concept | **UseCase** |
|---|---|
| Forme SysML |  |
| Forme ontologique |  |
| RDF/XML | <!– www.ex.com/req#UseCase1 –><br><owl :NamedIndividual<br>rdf :about="www.ex.com/req#UseCase1"><br><rdf :type<br>rdf :resource="www.ex.com/req#UseCase"/><br></owl :NamedIndividual> |

TABLE 3.9 – La transformation d'un cas d'utilisation

- **Transformation de la partie prenante :**
  La partie prenante est transformée aussi à un concept ontologique, dans l'exemple qui suit :

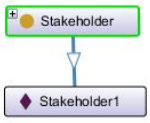
| Concept | **Stakeholder** |
|---|---|
| Forme SysML |  |
| Forme ontologique |  |
| RDF/XML | <!– www.ex.com/req#Stakeholder1 –><br><owl :NamedIndividual<br>rdf :about="www.ex.com/req#Stakeholder1"><br><rdf :type<br>rdf :resource="www.ex.com/req#Stakeholder"/><br></owl :NamedIndividual> |

TABLE 3.10 – La transformation de la partie prenante

- **Transformation du cas de test :**
  Un cas de teste (TestCase) peut étre un diagramme d'activité ou de séquence ou d'état.
  Il est transformé vers un concept ontologique ayant une relation de vérification avec le concept exigence transformé aussi.
  Dans le tableau suivant, il est montré un exemple de transformation d'un cas de teste (Diagramme de séquence).

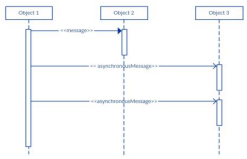
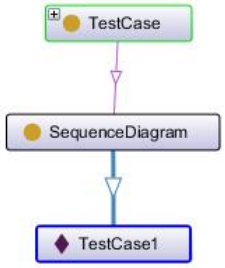
| Concept | **TestCase** |
|---|---|
| Forme SysML |  |
| Forme ontologique |  |
| RDF/XML | <!– www.ex.com/req#TestCase1 –><br><owl :NamedIndividual<br>rdf :about="www.ex.com/req#TestCase1"><br><rdf :type<br>rdf :resource="www.ex.com/req#SequenceDiagram"/><br></owl :NamedIndividual> |

TABLE 3.11 – La transformation du cas de test

- **Transformation des relations :**

  — La relation de raffinement : La relation " refine" relie une exigence avec un autre élément du modéle , il peut étre une autre exigence ou un cas d'utilisation , elle est transformée de sa forme SysML à une relation dans l'ontologie. L'exemple suivant montre la transformation d'une relation refine entre l'exigence fonctionnelle req1 et l'exigence non fonctionnelle req2 .

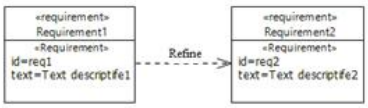
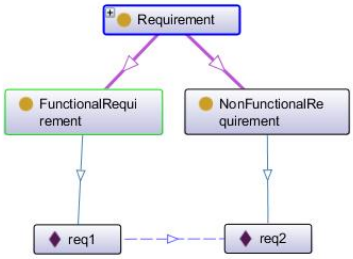
| Relation | **Refine** |
|---|---|
| Forme SysML |  |
| Forme ontologique |  |
| RDF/XML | <owl :NamedIndividual rdf :about="www.ex.com/req#req1"><br><rdf :type<br>rdf :resource="www.ex.com/req#FunctionalRequirement"/><br><Text></Text><br><Refine<br>rdf :resource="www.ex.com/req#req2"/><br></owl :NamedIndividual> |

TABLE 3.12 – La transformation de la relation de raffinement

— La relation de vérification : Le tableau suivant montre la transformation de la relation "verify" qui lie l'exigence fonctionnelle req1 avec le cas de test TestCase1, qui est un diagramme d'activité, de la forme SysML vers une forme ontologique :

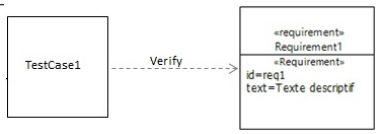
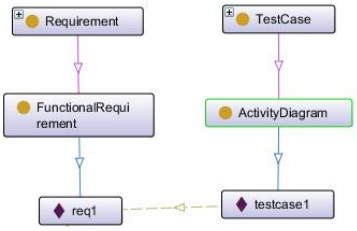
| Relation | **Verify** |
|---|---|
| Forme SysML |  |
| Forme ontologique |  |
| RDF/XML | <owl :NamedIndividual<br>rdf :about="www.ex.com/req#TestCase1"><br><rdf :type<br>rdf :resource="www.ex.com/req#TestCase"/><br><Text></Text><br><Verify rdf :resource="www.ex.com/req#req1"/><br></owl :NamedIndividual> |

TABLE 3.13 – La transformation de la raelation de vérification

— La relation de satisfaction : On prend dans ce tableau un exemple de transformation d'une relation de satisfaction " satisfy" entre le bloc Block1 est l'exigence fonctionnelle req1 de la forme SysML vers une relation entre deux concepts ontologiques ( Block1 et Req1) :

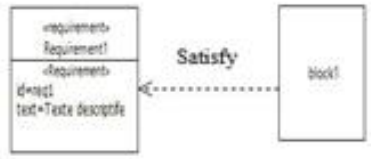
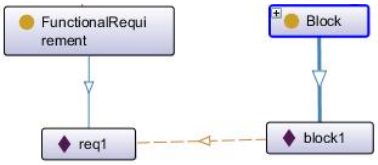
| Relation | **Satisfy** |
|---|---|
| Forme SysML |  |
| Forme ontologique |  |
| RDF/XML | <owl :NamedIndividual<br>rdf :about="www.ex.com/req#block1"><br><rdf :type rdf :resource="www.ex.com/req#Block"/><br><Text></Text><br><Satisy rdf :resource="www.ex.com/req#req1"/><br></owl :NamedIndividual> |

TABLE 3.14 – La transformation de la relation de satisfaction

Afin d'avoir une ontologie compléte , on fusionne la partie obtenue dans cette étape (la représentation ontologique des éléments du diagramme d'exigences SysML). On la donne le nom (ABOX) avec les concepts de l'ontologie définie qu'on considère comme partie fixe ( TBOX) . Ainsi, On pourra retirer des connaissances concernant le diagramme des exigences en analysant cette représentation qu'on aura aprés la fusion, en exécutant des requêtes .

La figure 3.5 montre ça :

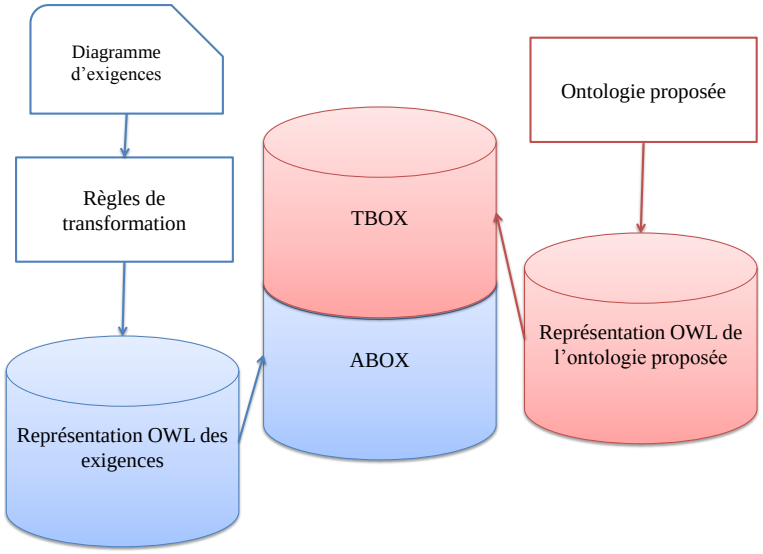



FIGURE 3.5 – Fusion des parties de l'ontologie

### 3.3.3 Analyse sémantique des spécifications SysML

Une fois nous obtenons une représentation ontologique des exigences SysML , l'analyste SysML peut procéder à la phase d'analyse . Ceci est possible en développant un framework qui permet à lui de poser des questions et avoir des réponses concernant la spécification SysML.
Afin de développer le framework , nous nous sommes basés sue le langage SPARQL (SPARQL Protocol and RDF Query Language) . C'est un langage et un protocole qui va donc permettre de faire des requêtes sur des données web et donc sur une ontologie. SPARQL est assez identique à SQL car on accède aussi aux données via ce langage. La différence est que SQL accède aux données d'une base de données et SPARQL accède aux données du Web des données[29] .
Ce langage de type SQL est donc utile pour interroger notre ontologie. Il permet plusieurs opérations comme SQL. Pour ce travail, seul les sélections nous intéresse car la création et modifications des données sont faites auparavant.
Lors des requêtes SPARQL, certaines spécificités d'OWL, de RDF et de RDFS ont été utilisées pour sélectionner ce qui nous intéresse, car l'ontologie est aussi composée d'attributs de RDF, RDFS et OWL. Pour pouvoir

utiliser toutes ces spécificités, il faut définir les espaces de noms avant la requête de sélection soit l'espace de nom de rdf, de rdfs et de l'ontologie.

```
PREFIX ont:<http://www.ontologies.com/monontologie.owl#>
PREFIX rdf:<http://www.w3.org/2000/01/rdf-shema#>
PREFIX rdfs:<http:www.w3.org/1999/02/rdf-ns-syntaxe#>
```

Le mot clé "PREFIX" permet d'en indiquer le préfixe puis l'adresse. Devant chaque propriété utilisée dans un triplet, on doit indiquer le préfixe suivi de la propriété comme suit : « rdf :proprieteRDF » ou « ont :proprieteOntologie».Ensuite comme en SQL, il faut projeter les ressources choisies avec "SELECT", puis choisir la ou les conditions avec des triplets en utilisant le mot clé "WHERE".

**Quelques requêtes exécutées :**

Nous proposons deux classes de requêtes : la premiére regroupe les requêtes basiques permettant d'interroger une représentation ontologique des exigences SysML, et la deuxiéme classe regroupe des requêtes qui sont relatives à une analyse particuliére ( impacts du changement , vérification , ... etc ).
Dans notre travail , on s'est limité aux questions relatives à la gestion de l'impact du changement .

- **Requêtes basiques :** Cette phase permet d'atteindre l'un des objectifs de l'ingénierie des exigences , c'est l'analyse des exigences tout au long du cycle de développement d'un système.
  Il faut lancer le processus d'analyse d'exigences en étudiant l'ensemble des exigences soumis par les parties prenantes pour les analyser, les compléter, les arranger, les tracer, et pour éliminer l'ambiguïté et la redondance entre les exigences .
  Notre démarche repose sur la proposition des requêtes SPARQL permettant de retirer des informations concernant les spécifications SysML, on a par exemple la requéte qui permet au concepteur ou à l'analyste de voir toutes les exigences d'un diagramme :

  ```
  SELECT ?X
     WHERE{
  ```

```
        ?X rdf:type ?Exigence
        ?Exigence rdf:type ?Diagramme
         Filter ?Exigence ns: Requirement
         }
```

Ou seulement les exeigences d'un type précis, prenons la requéte qui affiche toutes les exigences fonctionnelles :

```
SELECT ?X
   WHERE{
        ?X rdf:type ?ExigenceF
        ?ExigenceF rdf:type ?Exigence
         ?Exigence rdf:type ?Diagramme
         Filter ?Exigence ns: FunctionalRequirement
         }
```

ou toutes les exigences de processus :

```
SELECT ?X
   WHERE{
        ?X rdf:type ?ExigenceP
        ?ExigenceP rdf:type ?Exigence
         ?Exigence rdf:type ?Diagramme
         Filter ?Exigence ns: ProcessRequirement
         }
```

Comme il peut voir tous les blocs dans un diagramme, en exécutant la requête :

```
SELECT ?X
   WHERE{
        ?X rdf:type ?Bloc
        ?Bloc rdf:type ?Diagramme
         Filter ?Bloc ns: Block
         }
```

ou précisemment tous les blocs satisfaisant une exigence choisie :

```
SELECT ?X
   WHERE{
        ?X ns:satisfy ?Exigence
         }
```

Contrairement à la requéte ci-dessus, le concepteur peut savoir toutes les exigences satisfaites par un Bloc choisit , par la requête suivante :

```
SELECT ?X
   WHERE{
         ?Bloc ns:satisfy ?X
          }
```

Comme il peut afficher toutes les exigences dérivées d'une autre exigence :

```
SELECT ?X ?Y
   WHERE{
         ?X ns:Derived ?Y
          }
```

Ou tous les élèments raffinant une exigence :

```
SELECT ?X ?Y
   WHERE{
         ?X ns:Refine ?Y
          }
```

- **Requêtes complexes :** Parmi les objectifs de l'ingénierie des exigences , on a la modification des exigences et la détection de son impact sur les autres concepts .

  L'impact désigne la conséquence survenue d'un événement impactant sur le modèle. Ainsi, l'impact d'un événement impactant E sur un modèle M correspond à l'ensemble des éléments impactables de M marqués de l'étiquette « impacté » suite à la survenue de E.

  Les événements impactants se propagent sur les relations élémentaires pour se répercuter sur d'autres éléments impactables (cibles indirectes). Ils peuvent être de différents types : Ajout / Suppression / Modification.
  C'est l'une des applications que notre proposition peut apporter des solutions pour raisonner sur les exigences et déduire les impacts du changement. Cette fonctionnalité offre aux analystes la possibilité de faire une vérification et validation des diagrammes d'exigence.
  On prend un exemple de modification dans un diagramme , la figure

3.6 montre un diagramme d'exigence de la caméra mobile :

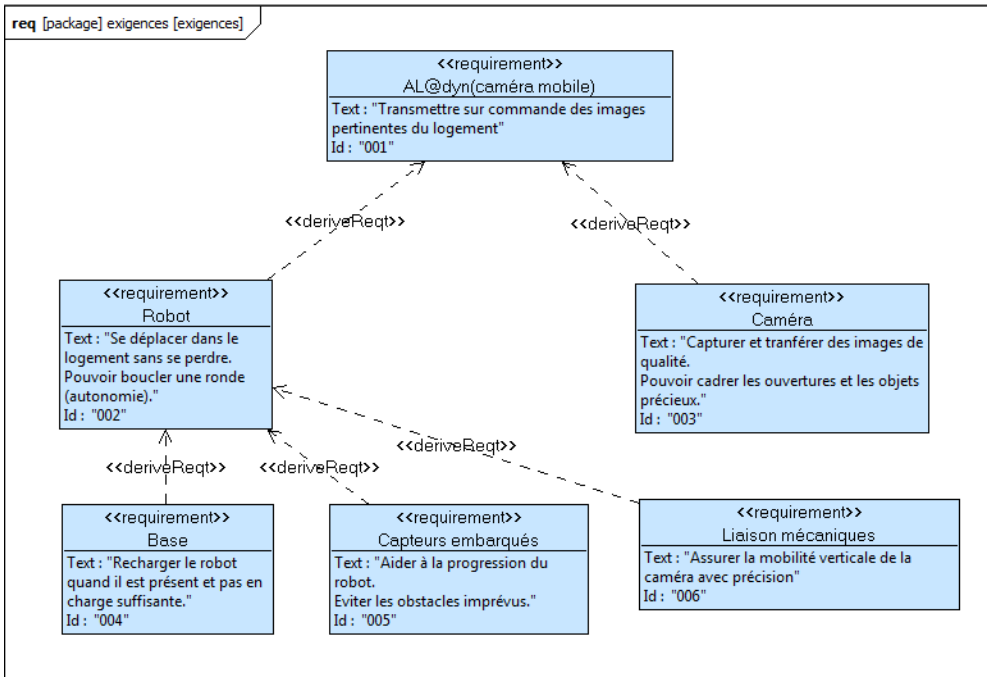



FIGURE 3.6 – Diagramme d'exigence de caméra mobile

Le changement d'une exigence peut avoir un impact sur : tous les éléments qui ont une relation directe ou indirecte avec elle .
Si on fait une modification dans la première exigence ( caméra mobile , id = 001 ) , on remarquera que les éléments suivants peuvent être impactés :
— Robot
— Caméra
— Base
— Capteurs embarqué
— Liaison mécanique .
Et si la modification est faite sur l'exigence (Robot, id=002) , les éléments qui peuvent être impactés sont :
— Base
— Capteurs embarqué
— Liaison mécanique .

Cela est réalisée en exécutant la requête suivante d'une façon récursive :

```
SELECT ?element
   WHERE{
         ?element ?relation ns:exigence
          }
```

Si on veut savoir les blocs impactés par le changement d'un bloc dans un diagramme d'exigence , on exécute la requéte :

```
SELECT ?exigence
   WHERE{
         ?Block ns:satisfy ?exigence
          }
```

Ce changement va avoir un impact sur tous les blocs satisfaisants une exigence impactée par ce changement, parce que l'impact du changement va se propager dans le diagramme.

Un changement d'une exigence touche aussi les cas de test. Les cas de test impactés sont tous les cas qui vérifient une exigence impactée par ce changement. La requéte qui les affiche est la suivante :

```
SELECT ?TestCase
   WHERE{
         ?TestCase ns:Verify ?exigence
          }
```

## 3.4 Conclusion

Dans ce chapitre, nous avons présenté l'approche que nous avons proposé pour l'analyse des spécifications SysML en se basant sur la notion des ontologies ( présentées en détails dans le deuxième chapitre ) . Nous avons expliqué les différentes étapes de notre démarche ,en commençant par la proposition d'une ontologie pour les exigences SysML et leurs transformation automatique pour aboutir à l'étape finale : l'analyse sémantique du diagramme donné .

# Chapitre 4

# Implémentation et étude de cas

## 4.1 Introduction

Après avoir effectué la conception d'une ontologie du domaine des exigences et d'un système d'affichage d'information qui exploite cette dernière, nous allons à présent entamer la réalisation de l'ontologie et du système d'affichage d'information que nous avons baptisé.
Notre objectif est de créer une ontologie décrivant les diagrammes d'exigence SysML afin de les donner de la sémantique , et exécuter quelques requétes pour retirer des connaissances d'elle .
Nous présentons dans ce chapitre l'ensemble des outils de développement utilisés et nous détaillons le processus de l'implémentation du systéme ainsi que la principale interface qui le compose à travers des fenêtres de capture.
Pour bien expliquer le principe de notre travail , on a présenté un étude de cas qui aide à comprendre l'objectif .

## 4.2 Outil d'analyse SysML2Owl

### 4.2.1 Module de transformation

Afin de faire la transformation du modéle des exigences vers le langage OWL, on a utilisé l'outil Acceleo, qui est un générateur de code source qui permet de mettre en œuvre l'approche MDA (Model driven architecture) pour réaliser des applications à partir de modèles basés sur EMF. La trans-

formation d'une exigence se fait via la régle suivante qui prend en entrée l'exigence SysML et donne en sortie sa forme ontologique :

```
<!-- http://www.semanticweb.org/onto#[m.id/] -->
    <owl:NamedIndividual rdf:about="http://www.semanticweb.org/onto#[m.id/]">
        <rdf:type rdf:resource="http://www.semanticweb.org#Requirement"/>
        <Text>[m.text/]</Text>
      </owl:NamedIndividual>
```

Et pour transformer ses relations avec les autres éléments , on utilise d'autres régles. Pour la transformation d'une relation de contenance, la régle suivante est utilisé :

```
[for(r2:Requirement |m.master)]
         <Contain rdf:resource="http://www.semanticweb.org/onto#[m.id/]"/>
        [/for]
```

Pour la transformation de la relation de dérivation " Derived" et son inverse "DerivedFrom", on utilise la règle :

```
 [comment]transforme a derive relation[/comment]
        [for(r:Requirement|m.derived)]
        <Derived rdf:resource="http://www.semanticweb.org/onto#[r.id/]"/>
        [/for]
        [comment]transforme a derivefrom relation[/comment]
        [for(r1:Requirement|m.derivedFrom)]
        <DerivedFrom rdf:resource="http://www.semanticweb.org/onto#[r1.id/]"/>
        [/for]
```

La régle qui suit fait la transformation de la relation de satisfaction entre deux éléments du modèle :

```
 [comment]transforme an satisfiedBy relation[/comment]
        [for(b:Block|m.satisfiedBy)]
        <SatisfiedBy rdf:resource="http://www.semanticweb.org/onto#[b.name/]"/>
        [/for]
```

### 4.2.2 Module sémantique

Ce module est basé sur l'exécution des requêtes SPARQL permettant d'interroger l'ontologie complète construite par le module de transformation.

Puisque notre ontologie est implémentée en langage OWL, or les fichiers OWL sont inexploitables en état brut car leur structure est très complexe. Donc pour pouvoir l'exploiter il nous a fallu un « traducteur » capable de traduire les balises et la sémantique véhiculée par le fichier OWL en objet manipulable par des programmes.

L'outil disponible qu'on a pu avoir est L'API JENA 3.7 . Cet outil est développé par une équipe de la firme HP (Hewlett Packard) dans le cadre du Projet HP « Labs Semantic Web Programme » qui a pour but de réaliser un outil d'exploitation des fichiers OWL. JENA est développé entièrement en Java, elle donne aux programmes la possibilité d'exploiter le contenu des fichiers RDF et OWL (extraction du contenu sémantique de ces derniers).

Le module sémantique donne la possibilité aux analystes SysML de raisonner sur les spécifications et analyser les exigences et leurs relations avec les autres éléments de modélisation.

Le framework développé permet d'exécuter des requêtes sémantiques (SPARQL) sur les spécifications SysML tout en cachant la complexité du langage SPARQL aux analystes SysML par une interface Homme/Machine qui n'est pas complexe en termes d'aspect visuel, et qui établit une communication entre l'analyste et la machine, et agit comme un interprète. Ce qui permet une meilleure manipulation de l'outil et une fiable analyse des exigences.

L'implémentation de l'outil SysML2OWL s'est basée sur le shéma de la figure 4.1 :

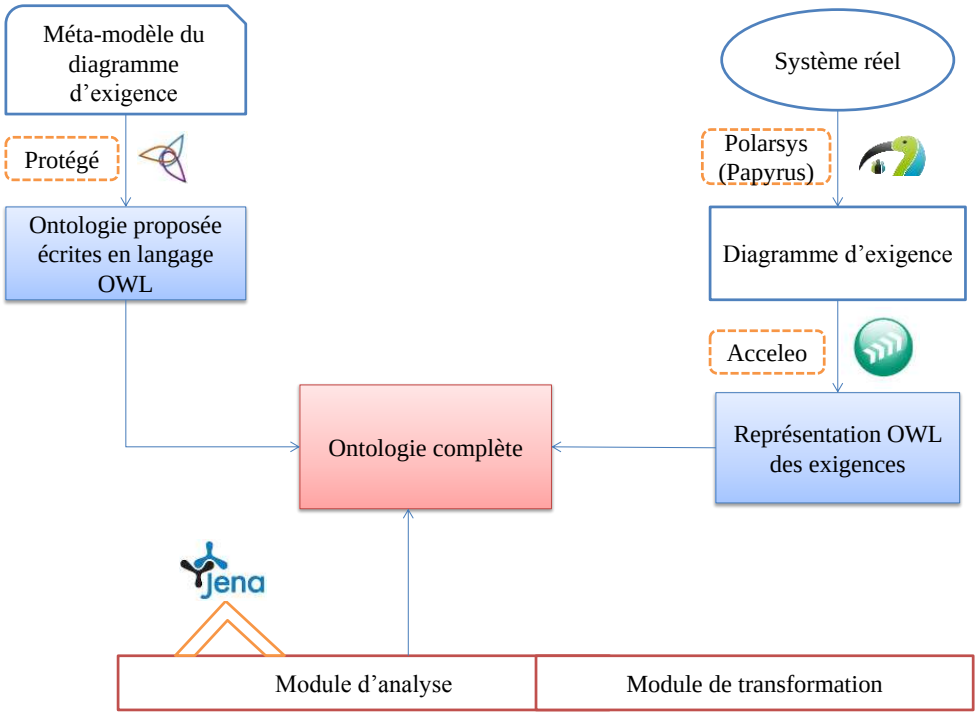



FIGURE 4.1 – Démarche générale

Ce schéma montre les étapes de l'implémentation de l'outil proposé, dont on a deux module , le premier est le module de transformation qui est réalisé en utilisant les outils Polarsys et Aceeleo, et le module sémantique qui repose sur le framework JENA.

## 4.3 Etude de cas

Notre étude de cas s'est portée sur le système de protection contre les incendies.

Les incendies peuvent avoir des conséquences dévastatrices, compromettre les biens et mettre en péril des vie humaines. C'est pourquoi il est est nécessaire de détecter une incendie le plus tôt possible.

Ce système repose sur trois aspects : Détection et confirmation de l'incendie, sonner l'alarme ensuite la suppression de l'incendie, ces exigences sont satisfaites par plusieurs éléments matériels , comme le système détecteur du feu et le système de suppression.

Le diagramme des exigences de ce système est représenté par la figure 4.2 :

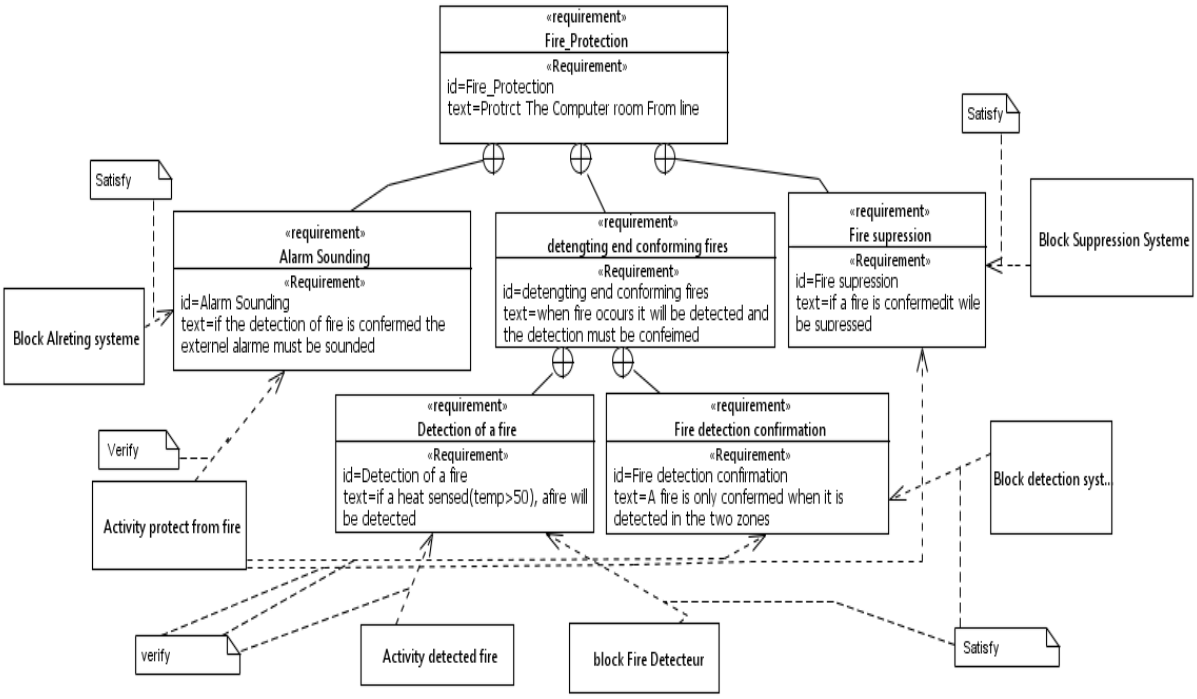



FIGURE 4.2 – Diagramme d'exigence du système de détection d'incendie

La transformation de ce diagramme en utilisant les régles d'acceleo citées a permis d'obtenir une ontologie compléte pour le systéme de détection d'incendie. Cette ontologie est montrée dans le graphe suivant :

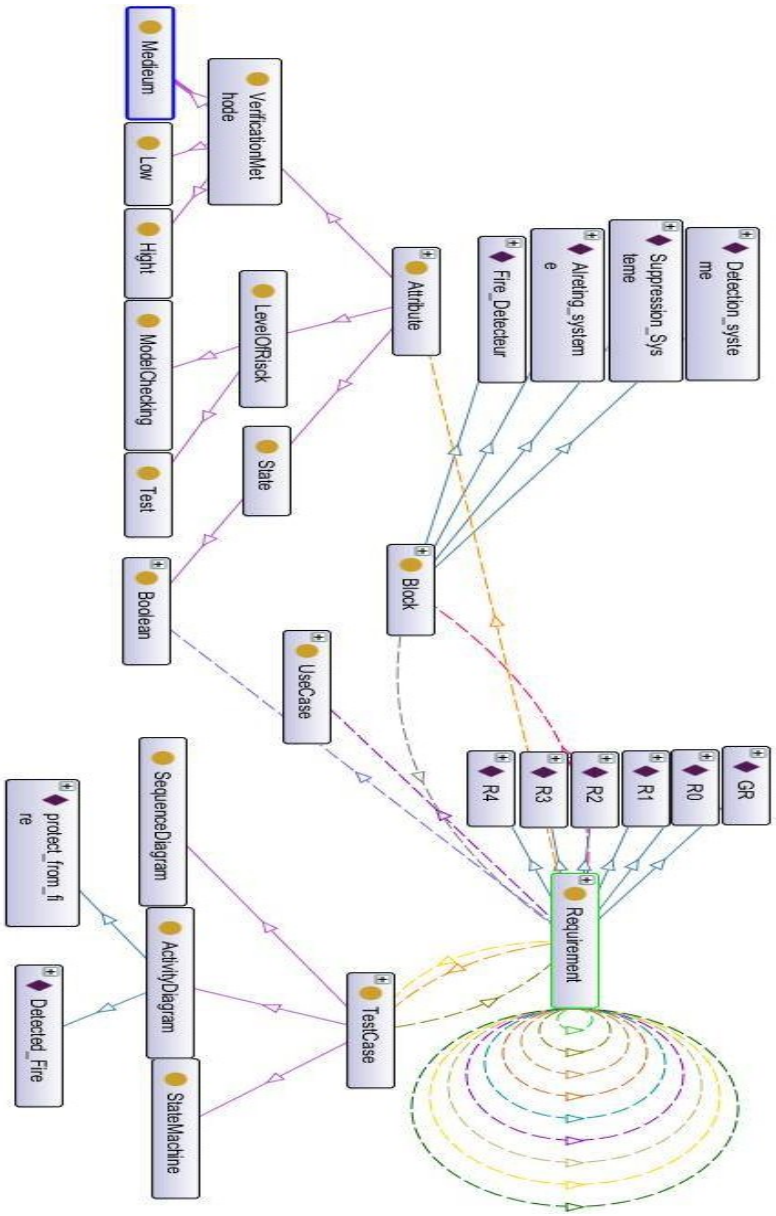



FIGURE 4.3 – Forme ontologique du système de détection d'incendie

On a pu voir le résultat aussi en RDF/XML, elle contient une partie de définition des concepts suivie des individus :

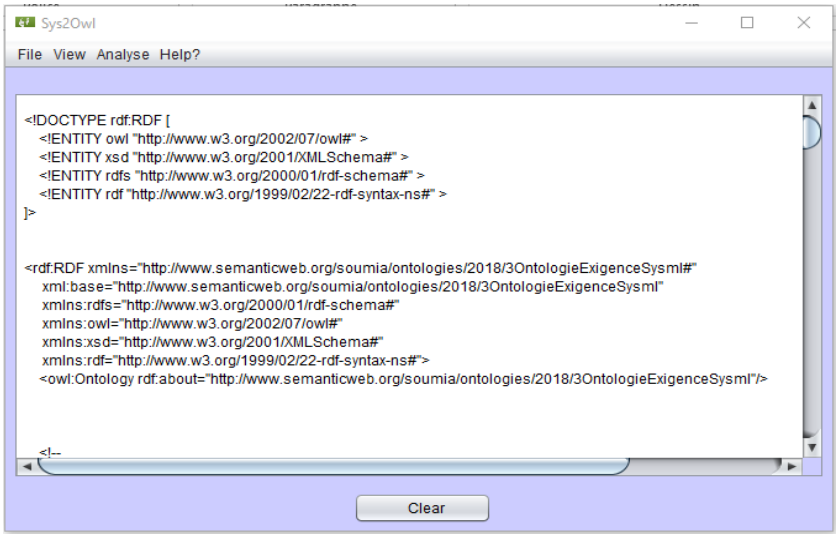



FIGURE 4.4 – La premiére partie du résultat de la transformation du système

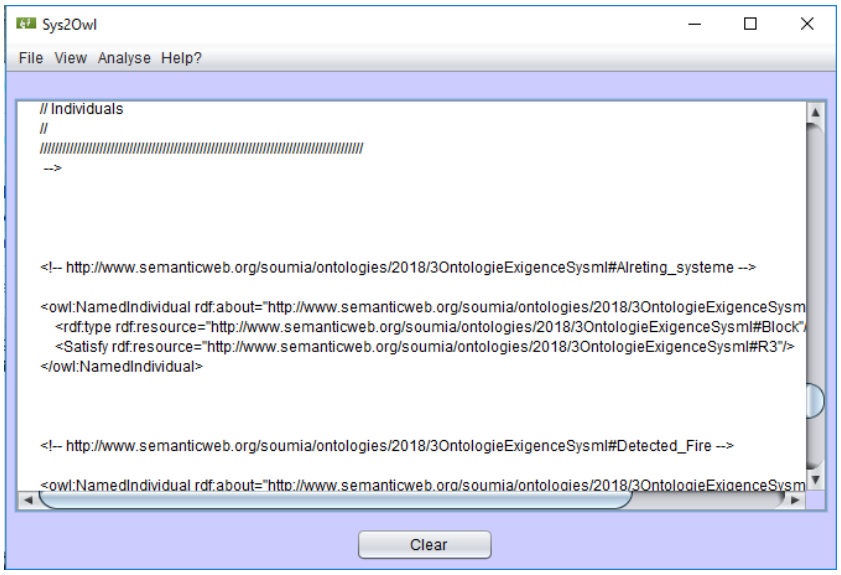



FIGURE 4.5 – La deuxième partie du résultat de la transformation du système

Venant à la phase de l'analyse, on a exécuté quelques requêtes pour afficher des informations concernant le diagramme.

On a voulu savoir toutes les exigences, pour faire cela, il suffit de cliquer sur la liste "Requirement" et choisir la requête voulue (All requirements) :

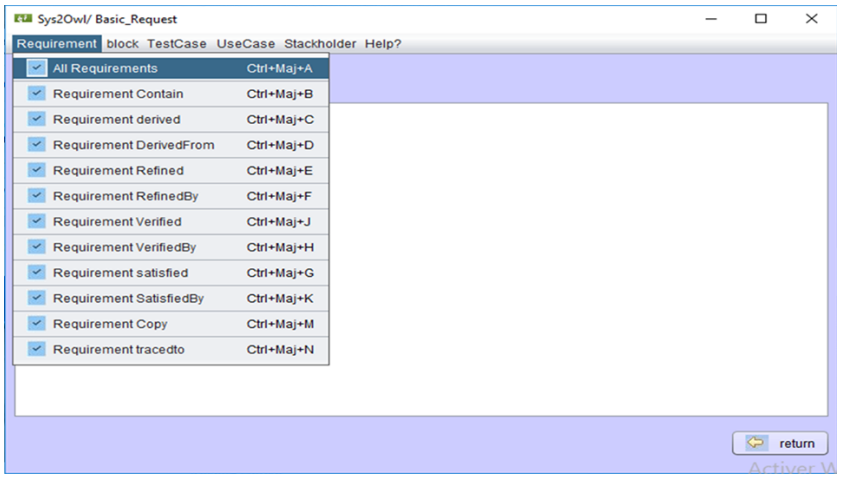



FIGURE 4.6 – Affichage de toutes les exigences du système

Le résultat est la suivante :

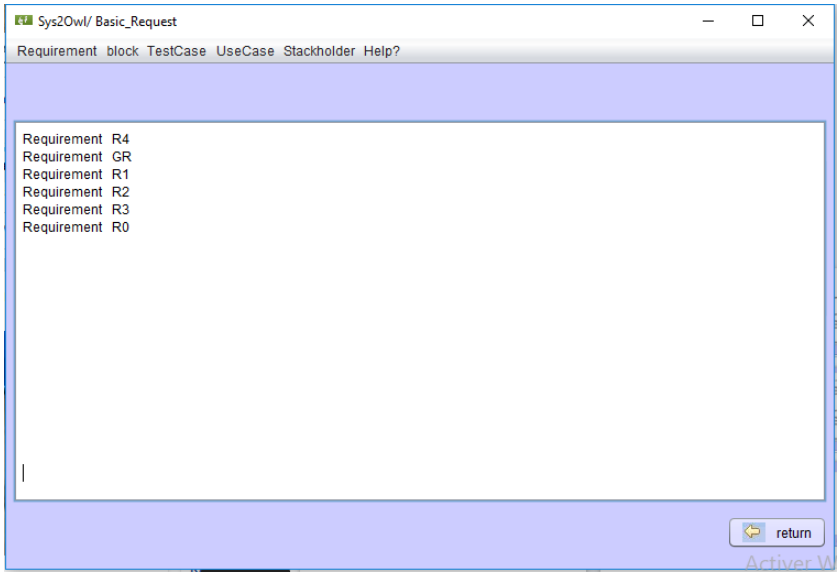



FIGURE 4.7 – Toutes les exigences du système

Aprés, on a essayé d'afficher toutes les exigences satisfaites par un block

choisi, pour arriver à ça, il faut cliquer sur la liste "Block" et choisir une requête :

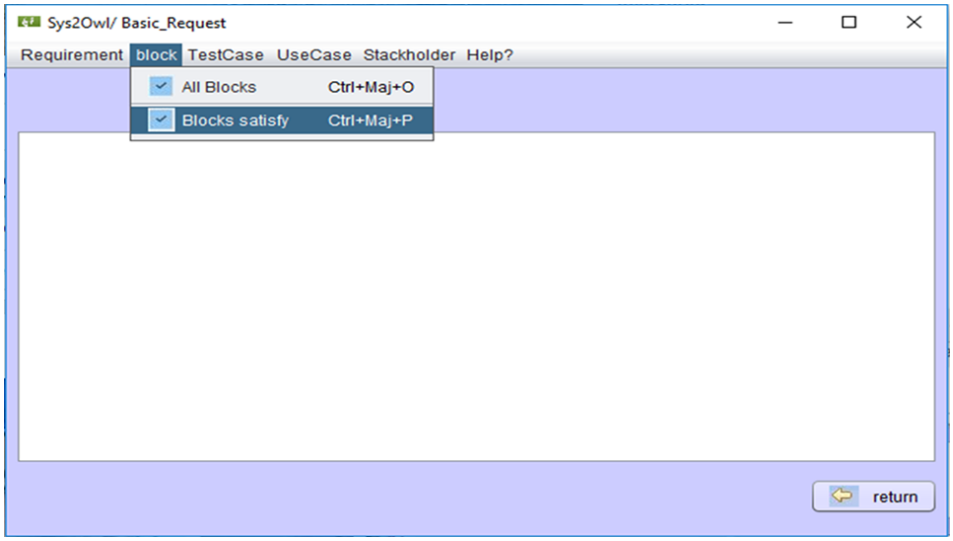



FIGURE 4.8 – Affichage de toutes les exigences satisfaites par un bloc

.

Ensuite, on doit choisir le bloc dont on veut savoir les exigences satisfaites par lui :

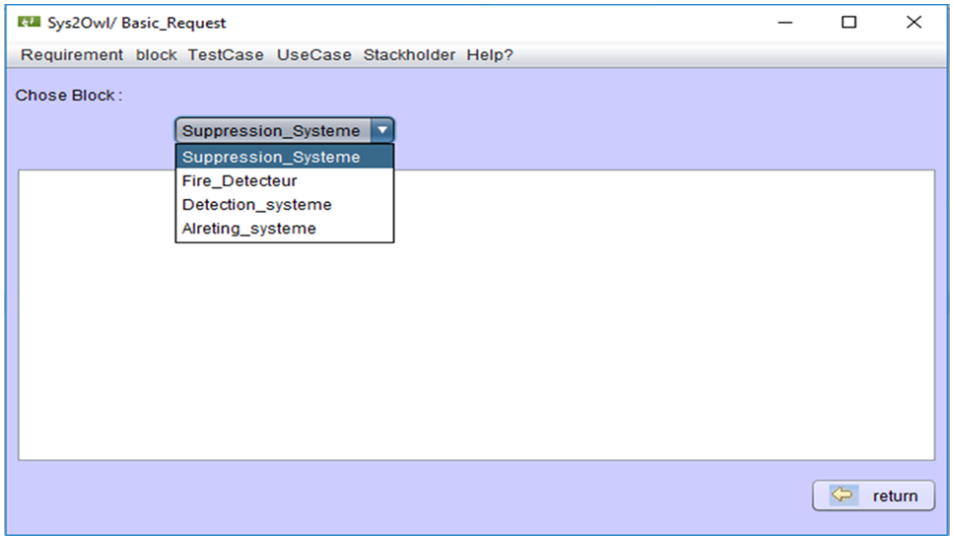



FIGURE 4.9 – Choix du bloc

Le résultat qui s'est affiché est :

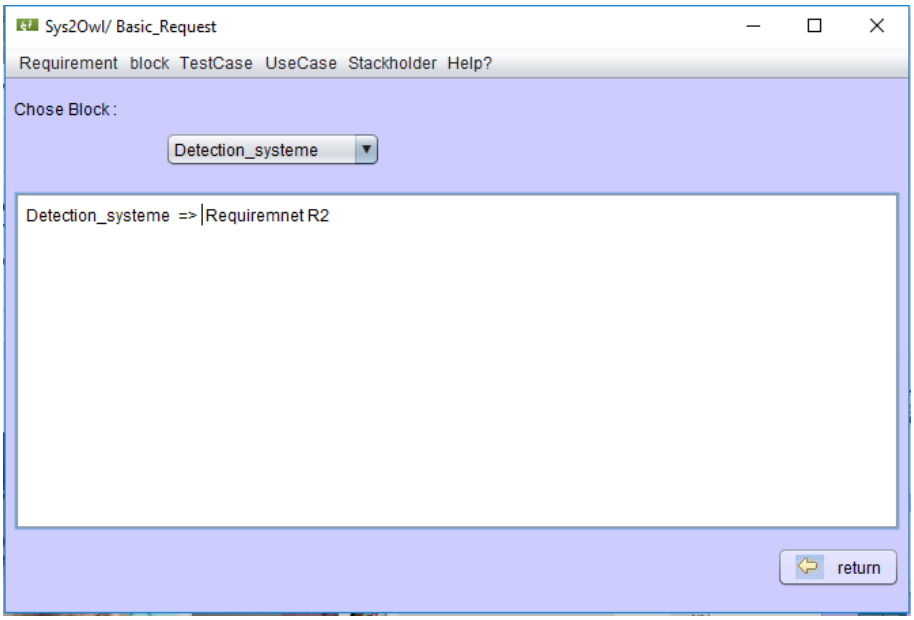



FIGURE 4.10 – Toutes les exigences satisfaites par un bloc

On a affichée aussi toutes les exigences que une exigence choisit contient :

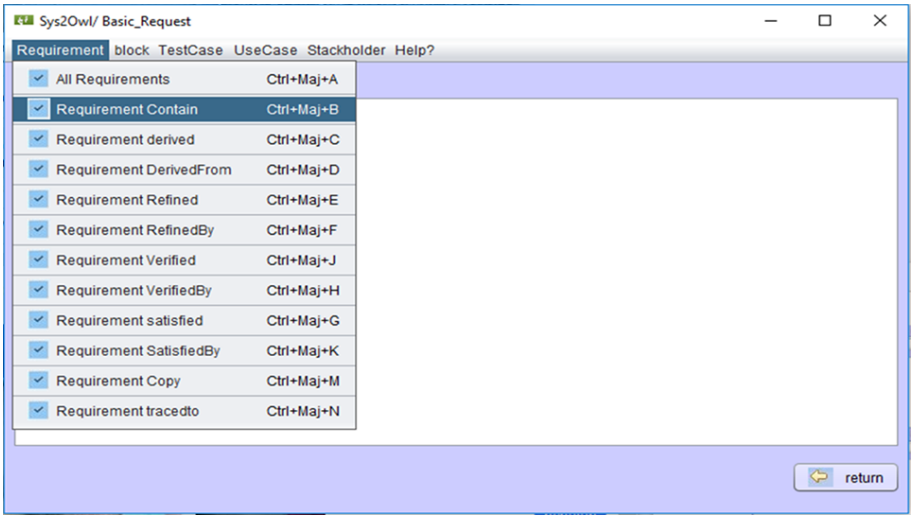



FIGURE 4.11 – Affichage de toutes les exigences contenues

et on choisit une des exigences du système :

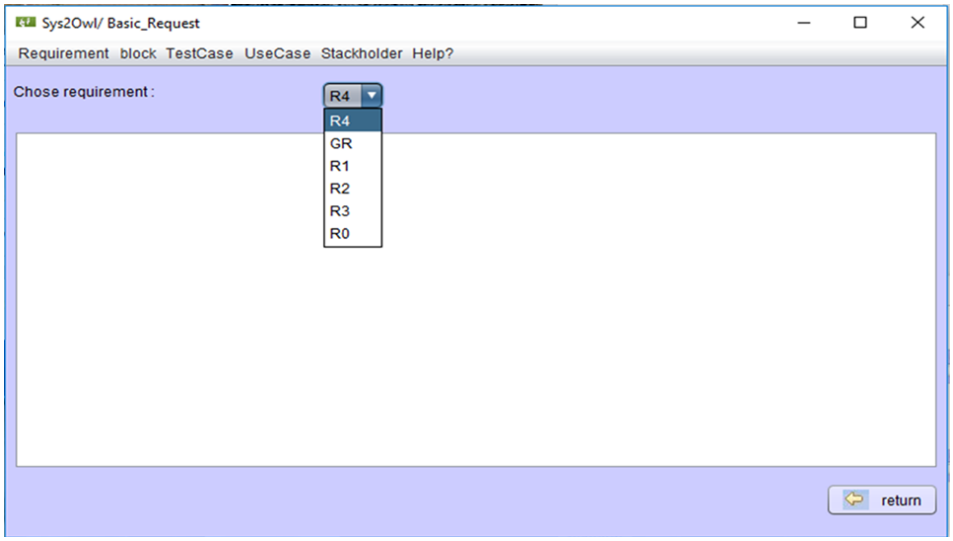



FIGURE 4.12 – Choix de l'exigence

Le résultat qui s'affiche est :

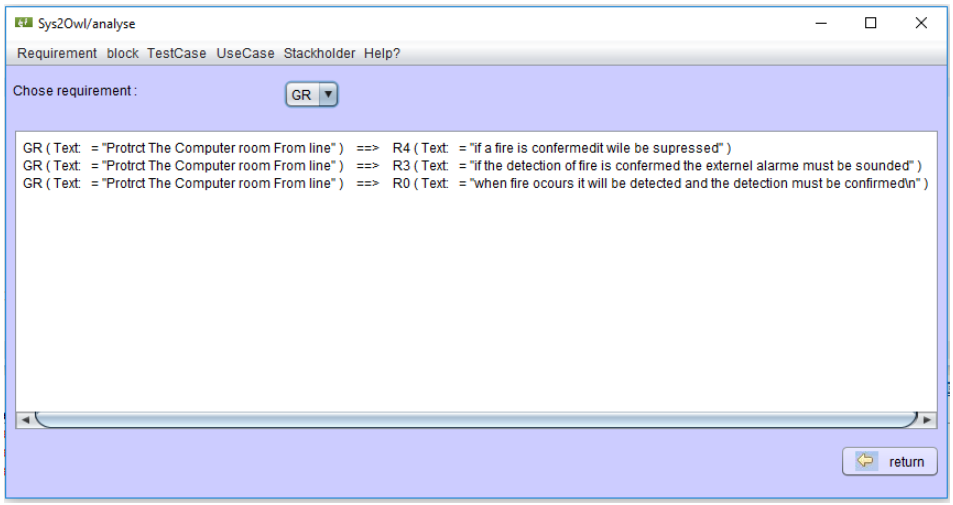



FIGURE 4.13 – Les exigences contenues dans l'exigence choisie

On arrive aux requêtes de l'impact du changement, on commence par le changement d'une exigence. On a cliqué sur la liste "Change Requirement" ensuite, choisir l'une des exigences :

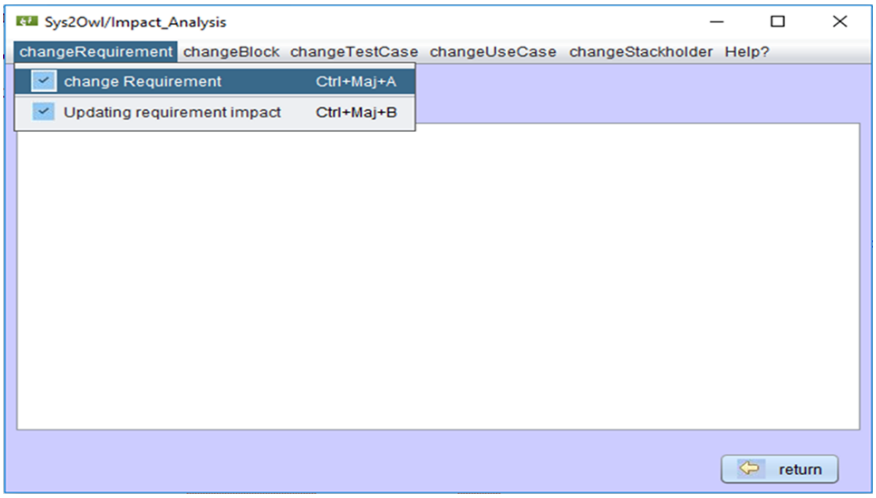



FIGURE 4.14 – Affichage de l'impact du changement d'une exigence

aprés le choix de l'exigence R3, on a eu le résultat :

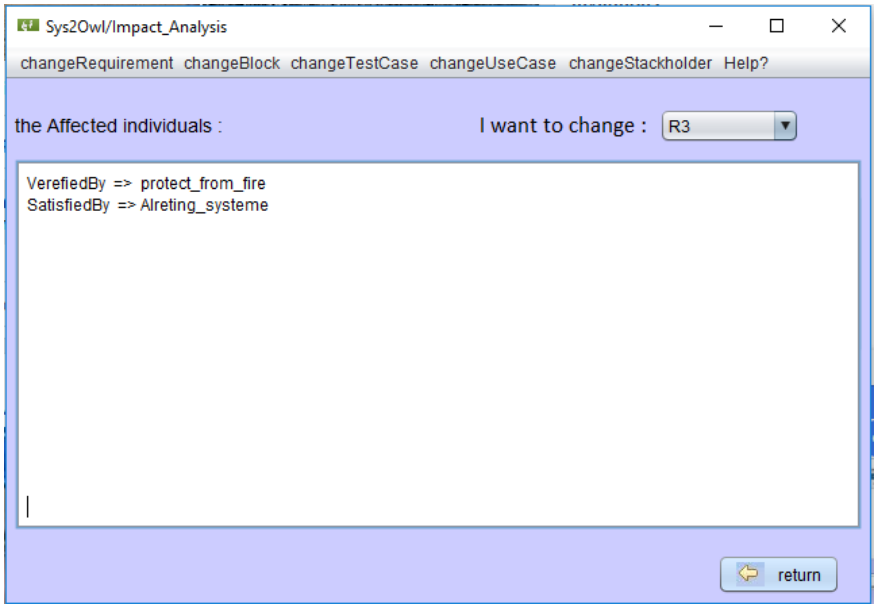



FIGURE 4.15 – L'impact du changement d'une exigence

Comme on a pu voir l'impact du changement d'un bloc choisie en cliquant sur la liste "Change Block" et choisir le bloc à changer :

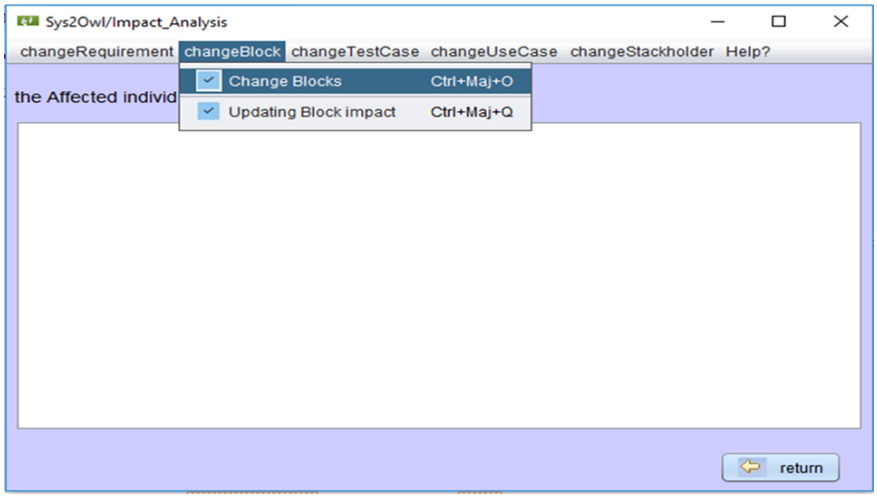



FIGURE 4.16 – Affichage de l'impact du changement d'un bloc

Le résultat est :

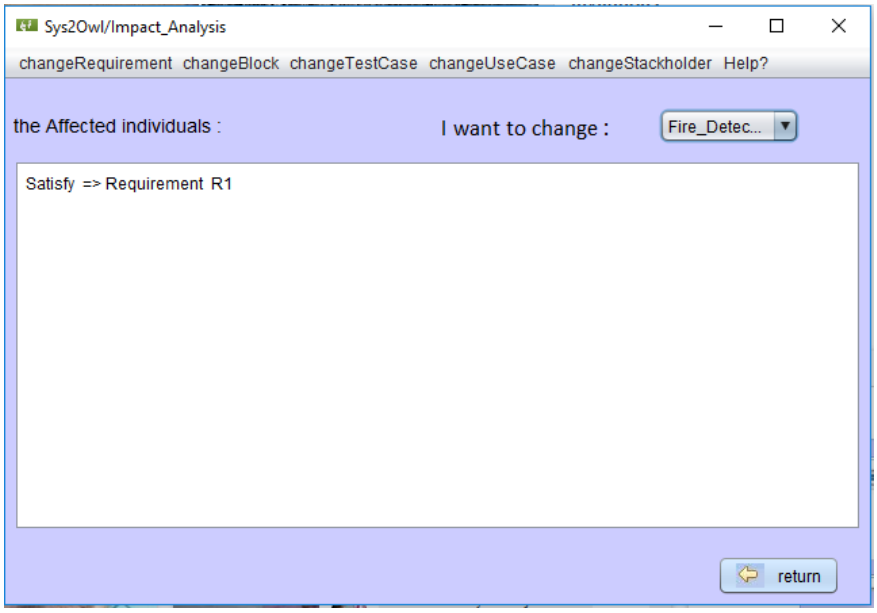



FIGURE 4.17 – Les exigences impactés par le changement du bloc

## 4.4 Conclusion

Dans ce chapitre nous avons présenté l'implémentation de notre système de manipulation d'ontologie. Nous avons tout d'abord présenté l'environnement de développement ainsi que les différents outils utilisés, et nous avons donné une description détaillée des étapes qu'on a suivi .

Après , on a donné des captures des interfaces qui sont conçues de manière à être conviviales et simples d'utilisation. Cette étape nous a aussi permis de nous familiariser avec les outils utilisés pour le développement d'ontologie .

# Conclusion générale

L'ingénierie des exigences est un processus qui a pour objet d'établir et de maintenir un accord avec les parties prenantes sur les exigences du système à construire dans le but de réaliser un système conforme au juste besoin. Parmi ses activités , on a : la spécification ,le développement ( l'analyse , la vérification et validation) et la gestion des exigences. L'analyse des exigences permet de collecter des données , donc comprendre le domaine et l'environnement et identifier les objectifs et les conflits.

Afin de permettre une meilleure analyse et gestion d'impact du changement , nous avons procédé à proposer une ontologie définissant les concepts du diagramme d'exigence , ce qui permet de donner une sémantique aux spécifications SysML et par conséquence pouvoir raisonner et inférer .

Ce mémoire porte sur l'investigation des problèmes et des insuffisances dans l'ingénierie des exigences et le développement des méthodes basées sur les ontologies pour améliorer l'exhaustivité, la cohérence et la qualité des connaissances retirées d'un diagramme d'exigences. le raisonnement ontologique a été appliqué pour identifier automatiquement les informations incohérentes ainsi que des défauts de qualité et de fournir à l'ingénieur des exigences des suggestions sur la façon de résoudre ces problèmes.

Donc ce travail contribue à :

- Faciliter le retirage des connaissances sur les exigences
- Automatiser la mesure de la connaissance des exigences concernant l'exhaustivité interne, cohérence et qualité.
- Permettre la gestion de l'impact des changements dans un diagramme d'exigence .

Certes le travail que nous avons fait n'était pas complétement parfait , mais il répond aux objectifs attendus . Sa réalisation nous a permis de se familiariser avec le langage SysML et la notion des ontologies , et même prendre une idée sur les outils Protégé et Acceleo .

Comme perspective, nous envisageons d'aller plus loin dans l'étape de l'analyse en proposant d'autres requêtes plus complexes. Le travail pourra être complété par un module de validation pour assurer une meilleure analyse des diagrammes d'exigences.

A la fin, nous espérons que ce modeste travail sera d'une aide certaine pour d'autres prochains travaux.

# Bibliographie


[1] HACINE GHERBI Ahcine. Construction d'une Ontologie pour le WEB Sémantique. Master's thesis, UNIVERSITE FERHAT ABBES –SETIF1-, 2015,2016.

[2] Ludovic Auxepaules. *Analyse des diagrammes de l apprenant dans un EIAH de la modélisation orientée objet.* PhD thesis, Université du Maine, 2009.

[3] Dougdag Samia Beldjerdi Meriem. Vers une vérification distribuée des diagrammes d activités SysML. Master's thesis, Université Dr Yahia Fares - Medea, 2015,2016.

[4] James A. Hendler Dean Allemang. *Semantic Web for the Working Ontologist, 2nd Edition.* Morgan Kaufmann, 2001.

[5] STEPHANE GASTON DIDIER FAGNON. Sysml : les diagrammes, 2012.

[6] ABDUL RAZZAQ et AL. Ontology for Attack detection : An intelligent Approach to web application Security. Avril 2014.

[7] Donghee Yoo et al. Ontology-based economics knowledge sharing System. 2013.

[8] Inès Gam El Golli. *Ingéniérie des exigences pour les systèmes d'information décisionnels : concepts, modèles et processus - la méthode CADWE.* Theses, Université Panthéon-Sorbonne - Paris I, October 2008.

[9] Fabien Gandon. Ontologies informatiques, Mai 2006. interstices.info.

[10] Veronique Giudicelli. Ontologies et l editeur protege, 2010.

[11] Thomas Holst. Structural Analysis of Unknown RDF Datasets via SPARQL Endpoints. Master's thesis, universite de Freie, Berlin, 2013.

[12] Marja Riitta Koivunen. Annotea and semantic web supported collaboration.

[13] TOUBAL SEGHIR Wafia Hassina LAMRI Abir. Vérification formelle des Diagrammes d'Activité SySML. Master's thesis, Université Dr Yahia Fares - Médéa, 2016 , 2017.

[14] Jean Pierre Lamy. *SysML un langage modele*. 2012.

[15] BOUDIA MALIKA. Transformation des diagrammes d états-transitions vers Maude. Master's thesis, Universite du Msila, 2007.

[16] Hiba Melhem. *Uses and applications of Semantic Web in Digital Libraries*. Theses, Université Grenoble Alpes, October 2017.

[17] RAHIM Messaoud. *Proposition d'une Démarche de Spécification et de Verification des Systèmes Complexes*. PhD thesis, Université des Sciences et de la Technologie Houari Boumediene, 2017.

[18] Alain Michard. *XML : langage et applications*. Eyrolles, 1999.

[19] Yang LI Amine MOKHTARI. Owl : Web ontology language, 2006.

[20] Raphael Faudou Nicolas Belloir, Jean-Michel Bruel. Modélisation des exigences en UML SysML. 2014.

[21] Sebastian Rudolph Pascal Hitzler, Markus Krotzsch. Knowledge representation for the semantic web owl 2, 2009.

[22] Nicolas Belloir Jean-Michel Bruel Natacha Hoang Congduc Pham. *Utilisation de SysML pour la modélisation des réseaux de capteurs*. PhD thesis, Université de Pau et des pays de l Adour.

[23] Samuel Rochet. *Formalization of System Engineering Processes : Proposal of a method to adapt generic processes to different application contexts*. Theses, Université Paul Sabatier - Toulouse III, November 2007.

[24] Pascal Roques. *SysMl par l exemple*. Eyrolles, 2009.

[25] Medieninf. Katja Siegemund. *Contributions To Ontology-Driven Requirements Engineering*. PhD thesis, Technischen Universität Dresden, 2014.

[26] Sanford Friedenthal Alan Moore Rick Steiner. *A pratical guide to SysML*. Morgan Kaufmann OMG Press, 2012.

[27] Skander Turki. *Model-Driven Engineering : Application oh the IEEE 15288 standard, of the MDA architecture and the SysML language to mechatronics systems design*. Theses, Université du Sud Toulon Var, October 2008.

[28] Arnaud Vandecasteele. *Expert knowledge ontological modeling for the analyse of abnormal behaviour : application for maritime surveillance.* Theses, Ecole Nationale Supérieure des Mines de Paris, October 2012.

[29] SAMUEL YERLY. Instanciation d'une "smart home" en owl et recherche des ses propriétés en sparql. Master's thesis, University of Fribourg (Switzerland), 2014.

[30] Zerarka Abderrahmane Zemir Abdenour. Vérification des exigences SysML sur les diagrammes d'activités. Master's thesis, Université Dr Yahia Fares, Médéa, 2015.