UNIVERSITE YAHIA FARES DE MEDEA

FACULTE DES SCIENCES

Département de Mathématique et d'Informatique



Mémoire de fin d'étude

Présenté pour l'obtention du Diplôme de Master en Informatique

Option : Ingénierie des Systèmes et Technologies du Web

THEME

Proposition d'une approche d'analyse de changements basée sur SysML et les ontologies et son application pour l'ingénierie des exigences

Proposé et dirigé par :

Mr. Messaoud RAHIM

Préparé et présenté par :

Melle. Kheira ROUMILI

Melle. Amina BOUKHELKHAL

Année Universitaire2018-2019

Remerciement

Il y a quelques années, après avoir choisi de terminer nos études dans cette discipline, nous ne pouvons pas deviner que notre projet de fin d'étude serait l'aboutissement d'un long chemin, une réalisation personnelle et une expérience humaine. C'est très agréable de comprendre, aujourd'hui, qu'une PFE est une réalisation collective, car sur ce chemin plusieurs personnes nous ont aidés et nous ont suivis. Sans ces personnes, nous ne pourrions pas le faire, et c'est à eux que nous voulons exprimer ici nos sincères remerciements.

Tout d'abord, grâce au Dieu tout puissant qui a nous avons donné la force et la volonté que nous avons achevé ce modeste travail

Puis, nous tenons à exprimer toute notre gratitude à notre encadreur Monsieur ***Messaoud RAHIM*** de nous avoir proposées ce sujet de fin d'étude, et pour nous avoir suivis, guidés, conseillés et aidés durant toute la période du travail.

Nous remercions également les membres du jury qui nous ont honorés d'avoir accepté d'examiner notre travail.

Nous remercions sincèrement les professeurs qui nous ont enseignée durant notre parcours universitaire à l'université « *YAHYA FARES* » Médéa.

Un grand merci à nos familles pour leur indéfectible soutien, leur patience et leur complaisance. Ce travail vous est dédicacé. Nous avons encore un merci particulier à tous nos amis pour leur prévenance et leurs encouragements, notamment ***Zahra, Rokia, Othmen, Haroune***. Merci d'avoir été là, et merci d'être là.

Dédicace

- Mes très chers parents qui m'ont toujours soutenu et aidé dans mes choix, et pour leurs vifs encouragements durant ces années d'études.
- Mes très chères sœurs et mes frères.
- Tous mes amis.
- Tous ceux qui me sont chers.



kheira

Dédicace

A tous ceux qui me sont chères.

A tous ceux qui m'aiment.

A tous ceux que j'aime.



amina

Résumé :

Le travail présenté dans ce mémoire vise à exploiter les ontologies et SysML pour contribuer à l'ingénierie des exigences et particulièrement l'analyse des exigences et l'analyse de l'impact des changements sur les spécifications lors des phases de développement des systèmes complexes. La démarche proposée consiste à transformer des spécifications SysML vers des ontologies pour pouvoir raisonner sur les changements et analyser l'impact de ces derniers sur les spécifications SysML. Un algorithme a été proposé pour la détermination des éléments de modélisation potentiellement impactés et le degré d'impact en se basant sur des mesures de similarité sémantiques. La démarche est évaluée en l'appliquant sur un cas d'étude modélisant un système complexe.

Mot clés : ontologie, OWL, Similarité Sémantique, SysML, Diagramme d'exigence, Diagramme d'activité, Diagramme de définition de bloc, impact de changement.

Summary:

The work presented in this report aims to exploiting ontologies and SysML to contribute to the requirements engineering and especially the requirements analysis and the analysis of the changes impact on the specifications during the development phases of the complex systems. The proposed approach consists of transforming SysML specifications into ontologies to be able to reason about changes and analyze their impact on SysML specifications. An algorithm has been proposed for the determination of potentially impacted modeling elements and degree of impact based on semantic similarity measures. The approach is evaluated by applying it to a case study modeling a complex system.

Keywords: Ontology, OWL, Semantics, SysML, Requirement Diagram, Activity Diagram, Block Definition Diagram, Change Impact.

ملخص:

يهدف العمل المقدم في هذه المذكرة إلى استغلال مزايا كل من الأنطولوجيا و SysML للمساهمة في هندسة المتطلبات وخاصة تحليل المتطلبات وتحليل تأثير التغييرات على المواصفات خلال مراحل تطوير النظم المعقدة. تتمثل الطريقة المقترحة في تحويل مواصفات SysML إلى أنطولوجيا لتكون قادرًا على استنتاج وتحليل تأثير التغييرات الطارئة على مواصفاتSysML . تم اقتراح خوارزمية لتحديد عناصر النمذجة التي يمكن أن تتأثر ودرجة التأثير بناءً على مقاييس التشابه الدلالي. يتم تقييم النهج من خلال تطبيقه على دراسة حالة نمذجة نظام معقد.

الكلمات المفتاحية : علم الوجود ، OWL، التشابه الدلالي ، SysML، مخطط المتطلبات ، مخطط النشاط ، مخطط تعريف كتلة ، تغيير الأثر.

Sommaire

Résumé

Sommaire

Liste des figures

Liste des Tableaux

Liste des Acronymes

Introduction Général ……... I-II

Chapitre I : Ingénierie de systèmes basée sur des modèles ……. 3-20

I.1 Introduction ……. 3

I.2 Ingénierie de Système ……...……...……... 3

I.3 Les modèles ……...……...……. 4

I.4 Ingénierie de systèmes basée sur des modèles ……...……...……... 5

I.5 Le Langage de modélisation SysML …….. 6

I.5.1 Les diagramme SysML ……... 7

I.5.1.1 Modélisation fonctionnelle ……. 9

I.5.1.2 Modélisation d'architecture (structure) ……..... 12

I.5.1.3 Modélisation dynamique (comportement) …….. 13

I.6 L'ingénierie dirigée par modèles ……...……. 16

I.7 Architecture dirigée par les modèles …….. 16

I.8 Outils MBSE ……...…… 19

I.9 Conclusion ……..... 20

Chapitre II : Représentation des Connaissances et raisonnement ……..…. 21-41

II.1 Introduction …… 21

II.2 Le Web sémantique ……..…… 21

II.3 Description des ressources RDF ……. 22

II.3.1 URI ……. 22

II.3.2 XML ……... 23

II.3.3 RDF …… 24

II.4 Description des ressources RDF-S ……...……..... 26

II.5 Représentation de données OWL ……...……. 29

II.5.1 La base de connaissances en logique descriptive ……...……... 31

II.5.2 Syntaxe et sémantique OWL ……...……. 31

II.6 Définition SPARQL ……...……. 36

II.7 Le langage SWRL ……....……...…….. 37

II.8 Outils du web sémantique …….…...……..... 39

II.9 Conclusion ……...……... 40

Chapitre III : Analyse de l'impact de changement ……... 41-46

III.1 Introduction ……... 41

III.2 Impact de changement ……...…… 42

III.3 Le processus d'analyse l'impact du changement ……..... 43

III.4 L'analyse de l'impact des changements dans le contexte SysML ……. 43

III.5 L'importance de la sémantique pour l'analyse d'impact du changement …….. 44

III.6 Similarité entre textes ……. 45

III.6.1 Similarité syntaxique …… 45

III.6.2 Similarité sémantique …….. 45

III.7 Conclusion ……. 47

Chapitre IV : Méthodologie et Conception …….. 47-67

IV.1 Introduction ……... 47

IV.2 La spécification de modélisation en SysML …… 47

IV.3 Architecture global d'approche proposée ……..…….. 49

IV.4 La Conception Détaillée ……...…….. 50

IV.4.1 Formalisation et Transformation …… 50

IV.4.1.1 La Formalisation ……. 50

IV.4.1.2 La Transformation …….. 55

IV.4.2 L'analyse sémantique des éléments de système ……... 63

IV.4.3 Degré d'impact ……... 65

IV.5 Conclusion ……. 67

Chapitre V : Implémentation et étude de cas ……... 68-84

V.1 Introduction …… 68

V.2 Implémentation ……... 68

V.2.1 Environnement de développement …….. 68

V.2.1.1 polarsys ……. 68

V.2.1.2 Netbeans (Java)…...……... 69

V.2.2 Les modules d'implémentions ……... 69

V.2.2.1 Transformation ……. 69

V.2.2.2 Analyse sémantique ……... 74

V.2.2.3 Impact de changement …… 74

V.3 Présentation de l'application …… 75

V.4 Etude de cas …… 76

V.4.1 Présentation de système ……... 76

V.4.2 Transformation de modèle ……. 81

V.4. 3 Analyse de système ……. 82

V.4. 4 Analyse de Impact de changement ……. 83

V.4. 5 Test et Résultat ……... 84

V.5 Conclusion …….. 85

Conclusion Général …… 86

Bibliographie et Webographie ……... 88

Annexes

Liste des figures

Chapitre I

Figure 1	Les vues modèle via MBSE	05
Figure 2	SysML en tant que profil UML	06
Figure 3	Présentation générale des diagrammes SysML	07
Figure 4	Les exigences de haut niveau se répercutent dans des détails spécifiques	10
Figure 5	Éléments d'un diagramme d'exigence	11
Figure 6	Diagramme Définition de Block	12
Figure 7	Diagramme d'activité	14
Figure 8	Quatre niveaux d'architecture de métamodèle	17
Figure 9	Modèles clés en MDA	17
Figure 10	Processus de transformation dans l'approche MDA	18

Chapitre II

Figure 1	Différentes briques technologiques du Web sémantique	21
Figure 2	URI syntaxe générique	22
Figure 3	Exemple de URI	23
Figure 4	Exemple de document XML	23
Figure 5	Exemple de syntaxe de XML namespaces	24
Figure 6	Exemple de triple RDF	25
Figure 7	Représentation graphique de RDF	25
Figure 8	Représentation XML de RDF	25
Figure 9	Représentation N-triples de RDF	26
Figure 10	Représentation turtle de RDF	26
Figure 11	Sous-langages OWL	29

Figure 12	Structure d'une requête SPARQL	36
Figure 13	Une simple requête et Répondre à cette requête par "graph matching"	37
Figure 14	La structure de rôle SWRL	38

Chapitre III

Figure 1	Processus de gestion d'impact de changement	43
Figure 2	Exemple d'une partie d'exigences et architecture d'un système	44

Chapitre IV

Figure 1	Les informations de traçabilité requises par les 3 diagrammes d'exigences, définition de Block et Activités	49
Figure 2	Schéma Synoptique de l'Approche Proposée	50
Figure 3	Les concepts de l'ontologie proposée pour représenter les diagrammes Sysml	53
Figure 4	La formalisation proposée de la relation "Satisfy"	53
Figure 5	Les relations de l'ontologie proposée pour représenter les diagrammes SysML	54
Figure 6	Le démarche de création de deux Box de connaissance OWL	55
Figure 7	La transformation du modèle SysML vers le langage OWL	55
Figure 8	Exemple avec relations déduites	65

Chapitre V

Figure 1	Composition d'un éditeur de diagramme	70
Figure 2	Vue d'ensemble Acceleo	70
Figure 3	Vue d'ensemble Templates	71
Figure 4	Aperçu du résultat attendu	74
Figure 5	Interface principale d'application	75
Figure 6	L'interface d'analyse de système	75
Figure 7	L'interface de mesure l'impact de changement	76
Figure 8	Système d'Airbag	76

Figure 9	Diagramme des exigences SysML	78
Figure 10	Diagramme BDD	79
Figure 11	Diagramme d'activité SysML	79
Figure 12	Interface graphique de sélection le modèle	81
Figure 13	Interface graphique de Transformation de modèle	81
Figure 14	Interface graphique de sélection Analyse de éléments d'un diagramme de Système	82
Figure 15	Interface graphique d'analyse de par élément / relation	82
Figure 16	Interface graphique d'analyse d'impact de suppression d'un élément	83
Figure 17	Interface graphique d'analyse d'impact de modification d'un élément sans détaille	83
Figure 18	Interface graphique d'analyse d'impact de modification d'un élément détaillée	84
Figure 19	Interface graphique d'analyse d'impact de modification filtrée par exigence	84
Figure 20	: mappage des éléments impact par le changement d'élément « Safety »	85

Liste des Tableaux

Chapitre I

Tableau 1	Comparatif des diagrammes UML et SysML	08
Tableau 2	Syntaxe de relation de diagramme des exigences	11
Tableau 3	Syntaxe de relation de BDD	13
Tableau 4	Syntaxe de relation de diagramme d'activité	15
Tableau 5	Outils MBSE les plus utilisés	19

Chapitre II

Tableau 1	Les classes RDF/RDF-S	27
Tableau 2	Les propriétés RDF/RDF-S	28
Tableau 3	Outils du Web sémantique les plus utilisés	39

Chapitre IV

Tableau 1	Transformation d'un Exigence	56
Tableau 2	Transformation d'bloc d'un BDD	57
Tableau 3	Transformation d'un diagramme d'activité	58
Tableau 4	Transformation d'un cas d'utilisateur	59
Tableau 5	Transformation d'un stakeholder	60
Tableau 6	Transformation de relation "satisfy" qui liée entre une exigence dans un diagramme d'exigence et un bloc dans un BDD	60
Tableau 7	Transformation de relation "verify" qui liée entre une exigence dans un diagramme d'exigence et un Activité	62

Chapitre V

Tableau 1	Résultat obtenu	85

Liste des Acronymes

CIM	**C**omputation **I**ndependent **M**odel
EJB	**E**nterprise **J**ava**B**eans
FOL	**F**irst-**O**rder **L**ogic
IBM	**I**nternational **B**usiness **M**achines Corporation
IDM	**I**ngénierie **D**irigée par les **M**odèles
INCOSE	**In**ternational **C**ouncil **o**n **S**ystem **E**ngineering
IRI	**I**nternational **R**essource **I**dentifier
IS	**I**ngénierie **S**ystème
KR	**K**nowledge **R**epresentation
MBSE	**M**odel-**B**ased **S**ystems **E**ngineering
MDE	**M**odel **D**riven **E**ngineering
MIC	**M**odel-**I**ntegrated **C**omputing
MIC	**M**odel-**I**ntegrated **C**omputing
MOF	**M**eta-**O**bject **F**acility
OMG	**O**bject **M**anagement **G**roup
OWL	**W**eb **O**ntology **L**anguage
OWL 2 EL	**OWL 2** for **E**xistential **L**ogic
OWL 2 QL	**OWL 2** for **Q**uery **L**anguage
OWL 2 RL	**OWL 2** for **R**ule **L**anguage
OWL DL	**OWL D**escription **L**ogics
PDM	**P**latform **D**escription **M**odel
PIM	**P**latform **I**ndependent **M**odel
PSM	**P**latform **S**pecific **M**odel

QVT	**Q**uery /**V**iew /**T**ransformation
RDF	**R**essource **D**escription **F**ramework
RDFS	**R**essource **D**escription **F**ramework **S**chema
RIF	**R**ule **I**nterchange **F**ormat
RuleML	**Rule M**arkup **L**anguage or **Rule M**odeling **L**anguage
SA/RT	**S**tructured **A**nalysis and **R**eal **T**ime
SADT	**S**tructured **A**nalysis and **D**esign **T**echnique
SDL	**S**ystem **D**efinition **L**anguage
SDLC	**S**ystems **D**evelopment **L**ife **C**ycle
SPARQL	**S**imple **P**rotocol **A**nd **R**df **Q**uery **L**anguage
SWRL	**S**emantic **W**eb **R**ule **L**anguage
SysML	**S**ystem **M**odeling **L**angage
UML	**U**nified **M**odeling **L**anguage
URI	**U**niversal **R**essource **I**dentifier
W3C	**W**orld **W**ide **W**eb **C**onsortium
XMI	**X**ML **M**etadata **I**nterchange
XML	E**X**tensible **M**arkup **L**anguage

Introduction Général

La complexité des systèmes a atteint un niveau sans précédent. Cela a créé de nouvelles opportunités, mais aussi à augmenter les défis pour les organisations qui créent et utilisent ses systèmes. Au motif que "le système est vu comme un ensemble composite de personnels, de matériels, de logiciels et de processus organisés pour que leur interfonctionnement permette d'atteindre une finalité bien déterminée" [21]. Sa démarche méthodologique est l'intérêt d'ingénierie de système qui inclut tous " les processus coopérative et interdisciplinaire qui englobe l'ensemble des activités adéquates pour concevoir, faire évoluer et vérifier un système apportant une solution économique et performante a des besoins spécifiques. " [2].

Bien que le langage modélisation UML permette de répondre à de nombreux besoins de l'Ingénierie Système (IS), il a été nécessaire d'adapter ce langage de modélisation basé sur UML pour l'IS. L'organisation internationale d'ingénierie système INCOSE (International Council on System Engineering) en 2001 a initié SysML (System Modeling Langage), qui remplace la modélisation de classes et d'objets par la modélisation de blocs pour englober tous les concepts logiciel, matériel, données, processus et même les personnes pour un vocabulaire plus adapté à l'ingénierie système. SysML est un langage semi-formel qui a avéré capable de représenter sous forme de diagramme les exigences, la structure et le comportement des systèmes, ce qui permet de relier différents éléments définissant la composition et les fonctionnalités du système souhaité.

Au cours de son cycle de développement du système, il est nécessaire de vérifier que la mise en œuvre du système final réponde aux exigences des parties prenantes (stakeholder). Le diagramme des exigences permet de restituer des éléments provenant d'autres schémas du modèle et ainsi de vérifier qu'une exigence a été prise en compte. Ainsi, une exigence ne peut pas être laissée seule, ce qui indiquerait qu'il manque quelque chose dans une vue modèle. En fin de compte, ils doivent tous être satisfaits. Ce concept de traçabilité peut être défini comme "la capacité à décrire et à suivre la durée d'une exigence dans les deux sens, vers ses origines ou vers sa mise en œuvre, en passant par toutes les spécifications associées" [32]

Représente les exigences d'un projet est une tâche cruciale étant donné que les exigences sont nombreuses et très variables. Le concept de traçabilité des exigences ne consiste pas uniquement de s'assurer de la cohérence entre ce que fait réellement le projet et ce qu'il doit faire. Il permet également de faciliter l'analyse d'impact en cas de changement sur la structure ainsi que le comportement de système.

En effet la modélisation SysML est semi-formelles, par conséquent, en cas de changement, schématiser la relation entre les exigences et la structure du système et son comportement ni pas assez efficace pour capturer et analyser l'impact de ce changement. Ainsi, une approche qui permet la représentation explicite des

connaissances et applique des mécanismes d'inférence s'avère d'une grande utilité. Il s'agit d'une démarche employant des langages d'ontologies. En particulier, le langage OWL est destiné à une représentation formelle des connaissances et il est supporté par des mécanismes d'inférence et d'interrogation. Cependant, OWL est destiné à une représentation formelle des connaissances et manque donc de modélisation compréhensible. [14]

Ce travail consiste à combiner la notation graphique attrayante de SysML et le raisonnement formel offert par OWL pour améliorer le processus d'ingénierie des exigences en effectuant des analyses sur les exigences et l'impact de changement sur les spécifications SysML. Nous proposons un passage automatique des modélisations SysML vers une représentation des connaissances via les ontologies. Par la suite, nous proposons des outils pour l'analyse des exigences et l'impact de changement sur la spécification SysML. L'analyse d'impact est améliorée par la suite, en proposant des mesures des degrés d'impact en se basant sur la similarité sémantique entre exigences et les éléments de la spécification SysML.

Ce travail est organisé en deux parties principales structurées comme suit :

La première partie présente un état de l'art et une étude détaillée sur le contexte de notre travail. Elle est composée de trois chapitres :

- Dans le premier chapitre nous introduisons l'ingénierie basée sur les modèles et le langage de modélisation SysML.
- Le deuxième chapitre est consacré à la présentation du web sémantique et les langages associés.
- Le troisième chapitre présente un état de l'art sur l'analyse de l'impact de changement des exigences.
- La deuxième partie est consacrée à notre conception et réalisation. Elle comporte deux chapitres :
- Dans le quatrième chapitre, nous montrons la méthodologie suivie et nous développons les étapes de conception.
- Dans la cinquième chapitre, nous expliquons la mise en œuvre de la démarche proposée et nous l'illustrons à travers une étude de cas.

Nous finissons notre mémoire par une conclusion générale et les perspectives ouvertes par ce travail de recherche.

Chapitre I

Ingénierie de systèmes basée sur des modèles

I.1 Introduction

Dans le développement de systèmes complexes et à grande échelle, faire automatiser ce processus permettent de gérer la complexité de ce système et de garantir que le système fourni répond à toutes les exigences. Une telle approche nécessite que les informations et les connaissances analysées soient capturées de manière formelle. L'ingénierie des systèmes basées sur des modèles (en anglais Model-base System Engineer -MBSE-). est l'application formalisée de la modélisation destinée à prendre en charge les activités requises pour le système, qui commence à la phase de conception et continue tout au long du développement et des phases ultérieures du cycle de vie.

SysML est un langage de modélisation graphique, créé comme norme internationale pour mettre en pratique l'approche MBSE La représentation graphique repose sur des diagrammes où chacun de ces diagrammes réponde à des besoins bien définis. Certains diagrammes s'intéressent à la structure du système et d'autres à son fonctionnement (comportemental). Un autre diagramme différent s'occupe de décrire les performances attendues.

Cette approche d'une ingénierie s'oriente vers le principe du « every thing is model ». Elle œuvre à fournir un cadre de développement dans lequel les modèles passent de l'état conceptuelle à l'état productif. En effet, l'approche vise à la séparation des concepts, en particulier la séparation de la technologie des concepts métiers. En pratique, pour un projet, cela permet d'accélérer le développement et de réduire les coûts de conception. Par conséquence, l'ingénierie dirigée par les modèles (IDM), ou Model Driven Engineering (MDE) en anglais a permis plusieurs améliorations significatives dans le développement des systèmes complexes en permettant de se concentrer sur une préoccupation plus abstraite que la programmation classique.

I.2 Ingénierie système

L'ingénierie système (IS) est une approche interdisciplinaire rassemblant tous les efforts techniques pour faire évoluer et vérifier un ensemble intégré de systèmes. Selon INCOSE [10], l'ingénierie des systèmes a pour mission "de créer et d'exécuter un processus interdisciplinaire afin de garantir que les besoins du client et des parties prenantes soient satisfaits de manière fiable, de haute qualité, rentable et dans les délais impartis tout au long du cycle de vie de développement du système" [36]. Cela signifie que l'ingénieur système est tenu de créer et de gérer un système répondant aux besoins des clients. Par conséquence l'ingénierie des

systèmes fournir une approche organisée, reproductible, itérative et convergente du développement de systèmes complexes.

L'ingénierie des systèmes concerne la conception, la construction et l'utilisation de systèmes composés d'entités concrètes ou de systèmes composés de processus. Car il propose une manière de penser à ces systèmes dans leurs contextes opérationnels. Cette façon de penser est au cœur de l'ingénierie des systèmes. Il commence par identifier les besoins des utilisateurs et des parties prenantes afin de s'assurer que le système répond au bon problème. Ensuit transforme ces exigences en une définition système, identifie les fonctions qui répondent à ces besoins, attribue ces fonctions aux entités système (composants) et confirme enfin que le système fonctionne comme prévu et qu'il répond à ses besoins.

C'est à la fois un processus technique et un processus de gestion. Le processus technique prend en charge l'analyse des besoins exprimés par les parties prenantes afin de décrire la fonctionnalité de système, en suit les efforts de conception et de mise en œuvre du système nécessaires pour transformer l'exigence opérationnelle en un système en décrivant la structure, et le comportement du système.

Le processus de gestion prend en charge le processus technique en planifiant, en évaluant les risques, en intégrant les différentes spécialités d'ingénierie et groupes de conception, en maintenant le contrôle de la configuration et en auditant en permanence les efforts visant à garantir le respect des objectifs de coût, de calendrier et de performance technique. Ensemble, la gestion et les processus techniques créent les systèmes qui répondront aux besoins des parties prenantes.

I.3 Les modèles

Les modèles sont communs à l'expérience humaine pour aider à comprendre le fonctionnement du monde. En ce sens, un modèle est une abstraction d'un système réel ou étudié [23]. Les modèles couvrent une vision allant de la forme à la fonction. Il existe des modèles tangibles et visibles qui imitent la forme mais non pas la fonction. À l'autre extrémité, il existe des modèles ou un lieu pour représenter visuellement la réalité, qui se préoccupe d'examiner le comportement de l'objet modélisé. Une caractéristique commune à de tels modèles est qu'ils ne capturent ou soulignent que certaines propriétés d'intérêt dans l'objet modélisé, tandis que la fidélité du modèle à l'objet réel est volontairement réduite ou limitée d'autres manières.

Dans le monde de la conception technique, "les modèles associent les idées qui sous-tendent une solution de conception avec ses relations de mise en œuvre et les connectent à la solution proposée ou au mécanisme existant qui résout le problème. Le modèle utilisant de cette manière est la pièce maîtresse de MBSE" [3].

Un tel modèle comporte quatre éléments.

- Le langage de définition de système (SDL) exprime et représente clairement sans ambiguïté afin de décrire le modèle de manière précise et compréhensible.
- La structure qui permet au modèle de capturer le comportement du système en décrivant clairement les relations entre les entités du système.
- L'argumentation car le modèle a pour objectif de représenter la conception du système de manière à ce que l'équipe de conception puisse démontrer que le système remplit les objectifs pour lesquels il a été conçu. Par conséquent, le modèle doit être en mesure de présenter un "argument" critique selon lequel le système répond aux exigences des parties prenantes.
- Présentation, non seulement le système doit être capable de présenter cet argument, mais il doit inclure un mécanisme permettant d'afficher ou de "présenter" l'argument de manière à pouvoir être vu et compris.

I.4 Ingénierie de systèmes basée sur des modèles

D'après l'INCOSE "L'ingénierie système basée sur les modèles (MBSE), est l'application formalisée de la modélisation permettant la gestion des exigences du système, la conception, les activités de vérification et de validation des étapes amont tout au long de processus de développement de système" [37]. Ces activités d'ingénierie des systèmes incluent, entre autres, l'analyse des besoins, l'analyse fonctionnelle, l'analyse de la performance, la conception de systèmes, les études commerciales, la spécification de l'architecture des systèmes et la vérification et la validation des systèmes.

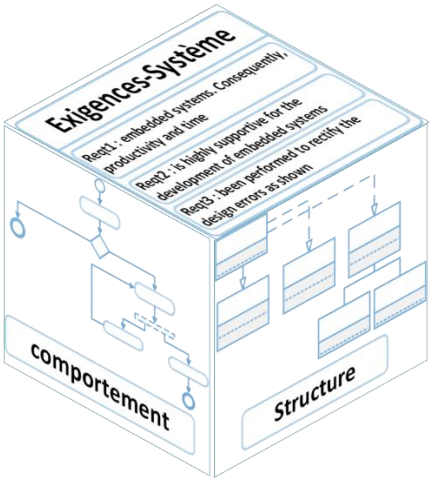


Figure I.1 : Les vues modèle via MBSE

Le but de MBSE est de créer et de gérer un seul modèle unifié du système pouvant représenter tous ses aspects variés, tels que les exigences, la structure et le comportement, comme indiqué dans la fig I.1. Le modèle de système est conceptualisé sous forme de graphe d'informations. Qui peut être visualisé ou projeté sur différentes perspectives, cette approche met en cohérence des modèles hétérogènes en se focalisant sur leur contenu. Elle permet de raisonner directement sur les éléments et leurs sémantiques. Les principes d'ingénierie des systèmes basés

sur des modèles continuent d'évoluer. Parmi les caractéristiques des approches MBSE :

- Mettre l'accent sur un "modèle" d'architecture système précis et complet à l'aide d'un cadre d'architecture avec plusieurs vues / points de vue, comme principal artefact de travail tout au long du cycle de développement du système (SDLC) ;
- Promouvoir l'utilisation de normes ouvertes pour la modélisation architecturale et l'interopérabilité des outils (SysML, UML2, XMI, …), pour spécifier le modèle d'architecture du système et servir de langage commun aux ingénieurs système et aux autres parties prenantes. ;
- Veiller à ce que le modèle d'architecture système soit guidé par les exigences dans la mesure où tous les éléments du modèle doivent être parfaitement traçables par rapport aux exigences du système et des utilisateurs ;
- S'assurer que les éléments du modèle doivent maintenir des relations d'intégrité structurelle et fonctionnelle et prendre en charge la traçabilité complète des dérivations dans toutes les vues et points de vue des parties prenantes du système.

Le terme Ingénierie de systèmes basée sur des modèles (MBSE) est populaire parmi les ingénieurs systèmes qui préconisent l'utilisation de SysML comme langage de modélisation d'architecture standard pour contrôler les différentes interfaces.

1.5 Le Langage de modélisation SysML

Les ingénieurs système utilisés une variété des techniques de modélisation. Parmi les plus connues, on trouve SADT et SA/RT, qui datent des années 80, ainsi que de nombreuses approches basées sur les réseaux de Pétri ou les machines à états finis. En 2003, l'INCOSE a décidé de faire d'UML ce langage commun pour l'IS. UML contenait en effet déjà à l'époque nombre de diagrammes indispensables, comme les diagrammes de séquence, d'états, de cas d'utilisation, etc.

Cependant, il restait une barrière psychologique importante à l'adoption d'UML par la communauté de l'IS sa teinture « logicielle » indélébile ! La possibilité d'extension d'UML, grâce au concept de stéréotype, a permis d'adapter le vocabulaire pour les ingénieurs système. En éliminant les mots « objet » et « classe » au profit du terme plus neutre « bloc », bref en gommant les aspects les plus informatiques d'UML, et en renommant ce langage de modélisation, l'OMG veut promouvoir SysML comme un nouveau langage différent d'UML, tout en profitant de sa filiation directe. [3]

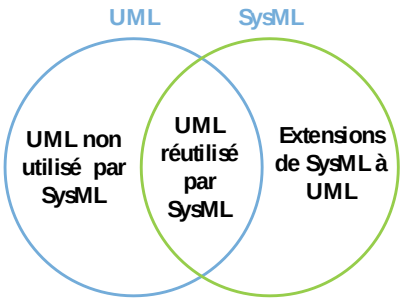


Figure I.2: SysML en tant que profil UML

SysML est un langage de modélisation graphique développé par l'OMG standardise depuis septembre 2007. Il est un profil d'UML 2.0 fournissant aux ingénieurs un langage de modélisation allant bien au-delà des problématiques de l'informatique.

La généralisation des concepts utilisés en UML enrichis de quelques notions donne SysML. Comme vous pouvez le constater sur la figure ci-dessous. Nous retrouvons des diagrammes de structure (diagrammes statiques) et des diagrammes de fonctionnement (diagrammes dynamiques) comme en UML.

Comme UML, SysML est un langage et non une méthode, par contre SysML aborde la notion de blocs qui deviendrons des parties mécaniques, électroniques, informatiques ou autres.

Les avantages d'une modélisation SysML sont :

- Partage des spécifications d'un système complexe entre tous les corps de métiers.
- Identification des risques et création d'une base d'analyse commune à tous les participants d'un projet.
- Facilite la gestion de projets complexes, l'évolutivité et la maintenabilité des systèmes complexes.
- Documente et capitalise le savoir de tous les corps de métiers dans un projet.

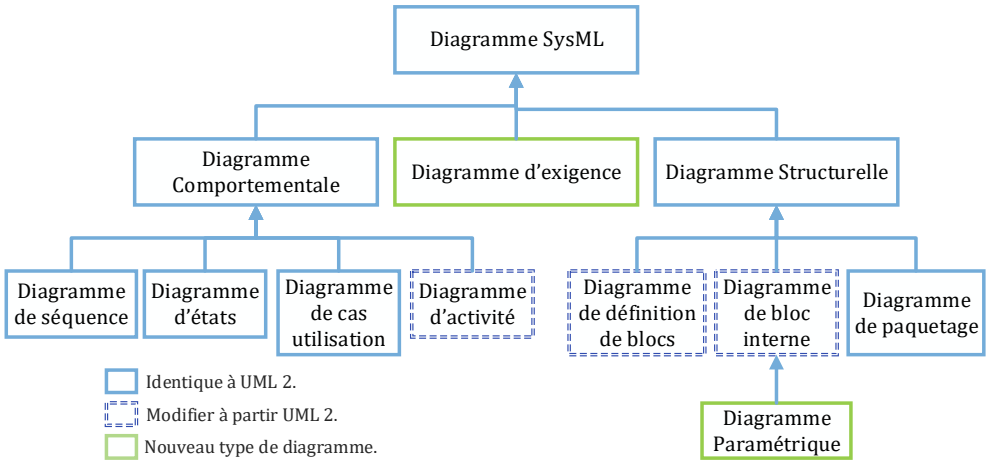


Figure I.3: Présentation générale des diagrammes SysML

I.5.1 Les diagramme SysML

Le passage de UML à SysML est très simple. Vous allez constater que les diagrammes sont moins nombreux et que SysML réutilise une bonne partie des diagrammes que vous connaissez déjà en UML.

Comme vous pouvez le vérifier ci-dessous SysML est très proche de UML et en même temps plus simple.

Tableau I.1 : Comparatif des diagrammes UML et SysML [6]

SysML	Description	UML
Use Case diagram	Identique en UML et en SysML, il modélise les fonctionnalités que le système doit fournir. Le cas d'utilisation est une unité fonctionnelle utilisée pour la description et la recette du système.	Use Case diagram
Sequence diagram	Identique en UML et en SysML le diagramme de séquence modélise la chronologie des interactions entre les éléments du système ou entre le système et l'extérieur.	Sequence diagram
Activity diagram	Même utilisation en UML et en SysML. Le diagramme d'activité modélise les flux d'informations et les flux d'activité du système.	Activity diagram
State Machine diagram	Identique en UML et en SysML, il représente les différents états que peut prendre un élément ou une opération ainsi que ses réactions aux évènements extérieurs.	State Machine diagram
Block Definition diagram	Le diagramme de Bloc en SysML est semblable au diagramme de Classe en UML. Il donne une représentation statique des entités du système, de leurs propriétés, de leurs opérations et de leurs opérations.	Class diagram
Internal Block diagram	Le diagramme internet de bloc SysML et le diagramme composite UML donnent une représentation « Boîte blanche » qui matérialise les imbrications des parties et leurs interconnections par les ports.	Composite Structure diagram
Package diagram	Le diagramme de Package montre l'organisation générale du modèle en UML comme en SysML. En SysML il sert en plus à donner différentes vues du système.	Package diagram
Parametric diagram	Nouveau dans SysML ce diagramme modélise les paramètres physiques du système. Il sert à tester les performances physiques et quantitatives du système.	N/A

Requirement diagram	Le diagramme de spécification est nouveau dans SysML et il permet de collecter et d'organiser toutes les exigences textuelles du système.	N/A
Allocation tables	Nouveau en SysML. Les tables d'allocation sont de simples tableaux et non des diagrammes qui récapitulent les spécifications afin de faciliter le suivi de projet.	N/A

Ainsi SysML permet de fournir un référentiel central qui supporte les analyses du système requises par l'IS, à savoir la décomposition fonctionnelle, les flux de données, et la décomposition structurelle.

Par ce qui suit, nous allons présenter un aperçu de chaque type de modélisation SysML, en commençant par le fonctionnement, suivi par la structure du système, et en terminant par le comportement du système (modélisation dynamique). Où pour chaque type de modélisation, nous allons montrer on détaille un de ses diagramme qui est utile dans notre conception par la suite

I.5.1.1 Modélisation fonctionnelle

SysML innove en proposant un diagramme qui permette de dessiner graphiquement au but de mettre en œuvre l'analyse d'un système complexe. En commençant par les exigences système, sous-système ou même équipement. Aussi le mettre en œuvre des techniques via des cas d'utilisation, ainsi que tous l'usage possible de système via diagramme de séquence.

- Diagramme d'exigences est une spécificité SysML pour que les exigences soient parties prenantes du référentiel de modélisation et puissent ensuite être reliées aux éléments structurels et dynamiques.
- Diagramme de cas d'utilisation est la mise en œuvre de la technique des cas d'utilisation, très utilisée avec UML.
- Diagramme de séquence est un diagramme très pratique pour représenter les scénarios d'utilisation du système par ses acteurs.

Diagramme des exigences :

Une exigence énonce une capacité ou une condition qu'un système doit satisfaire. Elle peut spécifier une fonction qu'un système doit exécuter ou une condition de performance qu'un système doit remplir, Les exigences servent à établir un contrat entre le client et les réalisateurs du futur système. [13]

La création d'exigences est une tâche complexe car elle inclut un ensemble de processus tels que la sollicitation, l'analyse, la spécification, la validation et la gestion. Ils sont classés en trois types d'exigences

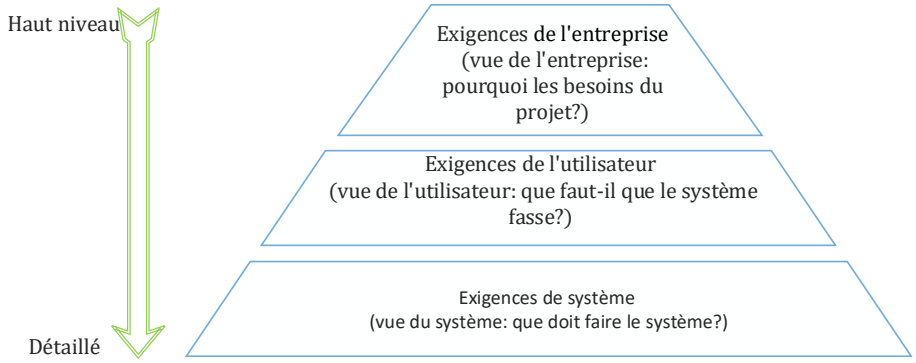


Figure I.4: Les exigences de haut niveau se répercutent dans des détails spécifiques [9]

Les exigences de l'entreprise incluent des déclarations de haut niveau sur les buts, objectifs. Par la suit les exigences des utilisateurs autrement dit les besoins des groupes d'utilisateurs distincts pour définir ce qu'ils attendent d'un système (solution) particulière. Et au niveau inférieur plus détaillée les exigences de système ou les exigences de la solution décrivent les caractéristiques qu'un produit doit posséder pour répondre aux besoins des utilisateurs et de l'entreprise elle-même.

Les modélisateurs ont également possible de faire la distinction entre différentes catégories des exigences par exemple les exigences opérationnelles, performances, ou d'interfaces qui on peut les regroupe ou deux classes principale les exigences fonctionnelles et les exigences non fonctionnelles

Les exigences fonctionnelles décrivent le comportement d'un produit, ses caractéristiques et ses fonctions. Alors que, Les exigences non fonctionnelles décrivent les caractéristiques générales d'un système. Ils sont également appelés attributs de qualité.

La représentation graphique de diagramme de exigences est comme suit, chaque exigence est décrite par un texte à l'intérieur d'un rectangle marqué du stéréotype « requirement » identifié de façon unique pour des raisons de traçabilité. Une exigence peut être décomposée en sous-exigences, de sorte que plusieurs exigences puissent être organisées en un arbre d'exigences composées. Les exigences peuvent être liées les unes aux autres, ainsi qu'aux éléments d'analyse, de conception, de mise en œuvre et de test. [12]

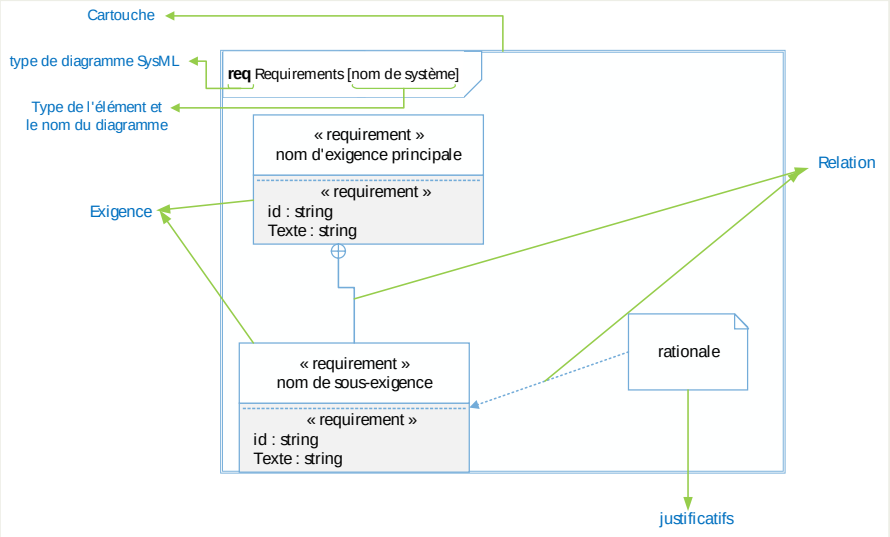


Figure I.5: éléments d'un diagramme d'exigence

La syntaxe des relations de diagramme d'exigence

Tableau I.2 : syntaxe de relation de diagramme des exigences

	Inclusion : B contient A (la satisfaction de A est nécessaire à celle de B)
	Dérivation : A a été identifiée comme un moyen de satisfaire B
	Raffinement : A raffine B (B concerne A)
	Satisfaction : les propriétés réalisées de A sont conformes à B
	Vérification : A permet de déterminer si B est réalisée ou non
	Traçabilité : A est à l'origine de B

SysML permet d'utiliser des notes graphiques sur tous les types de diagrammes. Deux mots-clés particuliers ont été définis afin de représenter :

- Des problèmes à résoudre (« problem ») ;
- Des justificatifs (« rationale »).

I.5.1.2 Modélisation d'architecture (structure)

La modélisation structurelle par SysML proposé utiliser le concept universel de « block » pour modéliser tout élément structure, ainsi que les deux types de diagrammes associés. Et structurer un modèle en packages. Diagramme de définition de blocs est pour définir tous les types d'éléments structurels du modèle système, sous-systèmes, blocs, types de données, unités, etc.

- Diagramme interne de bloc est pour expliciter la décomposition structurelle des blocs complexes en termes de sous-blocs, ports, flots et connecteurs.
- Diagramme de packages pour structurer notre modèle à des fins de travail en équipe ou de réutilisation.
- Diagramme paramétrique est une nouveauté SysML qui redéfinit le diagramme interne de bloc SysML, et permet d'intégrer des analyses systèmes (performance, fiabilité, etc.) avec des blocs de contrainte.

Diagramme de définition de blocs :

Le diagramme de définition de bloc (BDD) est utilisé pour définir les caractéristiques structurelles ou bien entités physiques d'un système, ainsi que de relations entre ces entités.

Le BDD répertorie les constituants du système sous forme de block en précisant leur rôle et leur quantité. Chaque block 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.

Dans un bdd, un bloc est représenté graphiquement par un rectangle marqué par défaut par de stéréotype « block » mais il est possible de définissons un nouvelle stéréotype tels que « system », « subsystem », etc. Tous les autres composants ont des labels indiquant ce qu'ils contiennent comme valeurs par exemple parties. Il existe également quatre relations qui permette de relier entre les différents blocks de BDD.

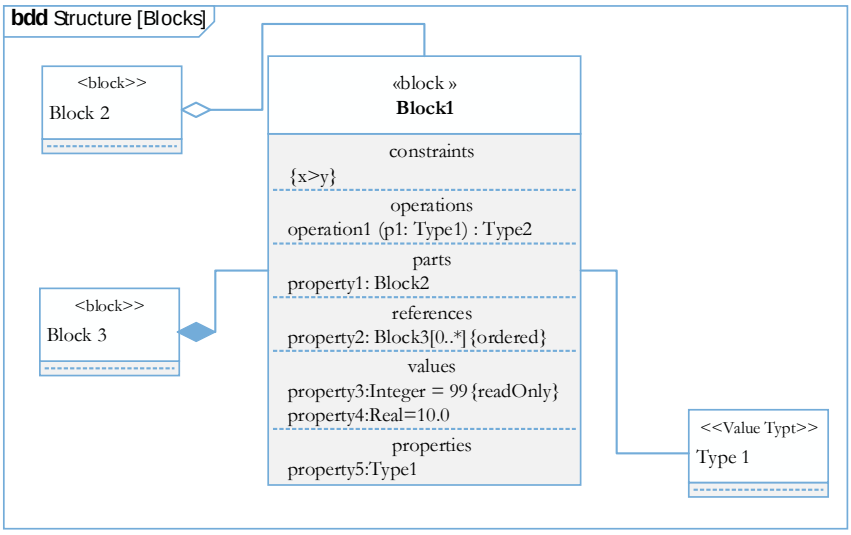
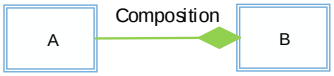
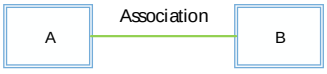
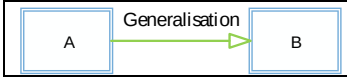


Figure I.6 : Diagramme Définition de Block

Les propriétés sont les caractéristiques structurelles de base des blocs. Les principaux uns sont :

- les constantes(constraints)est Une contrainte est simplement une condition portant sur un ou plusieurs éléments du modèle qui doit être vérifiée par les éléments concernés ;
- les parties (part properties) décrivent la hiérarchie de décomposition du bloc en termes d'autres blocs ;
- les valeurs (value properties) décrivent des caractéristiques quantifiables en terme de value types (domaine de valeur, dimension et unité optionnelles).

Tableau I.3 : syntaxe de relation de BDD

	A entre dans la composition de B et lui est indispensable (B est une composante obligatoire de A)
	A entre dans la composition de B sans être indispensable à son fonctionnement
	Une association représente une relation sémantique durable entre deux blocs
	A est une sorte de B

Multiplicités d'association signifie qu'à chaque extrémité d'une association, une indication de multiplicité doit être incluse. Il spécifie sous forme d'intervalle le nombre d'instances pouvant participer à une relation avec une instance de l'autre bloc dans le cadre d'une association. Une instance est une copie d'un certain bloc avec sa propre identité.

I .5.1.3 Modélisation dynamique (comportement)

En ce qui concerne la modélisation dynamique. Les constructions comportementales dynamiques utilisées dans les diagrammes comportementaux SysML incluent le diagramme d'activité et le diagramme d'état.

- Diagramme d'activité est une extension des activités UML 2, sert à modéliser avec précision le flux de contrôle et le flux d'entrée ainsi que les résultats parmi les actions.
- Diagramme d'états réalise la modélisation des modes de fonctionnement d'un système, et spécifie un comportement basé sur l'état en termes d'états du système et de leurs transitions, ainsi que le cycle de vie des principaux blocs constitutifs.

Diagramme d'activité

Le diagramme d'activité est l'un des diagrammes dynamiques proposés par SysML.il est utilisé pour représenter les étapes d'un traitement et les flots de données et de contrôle entre les actions. Il est utilisé majoritairement pour l'expression de la logique de contrôle et d'entrées/sorties. Ainsi qu'il propose des capacités supplémentaires importantes comme celle de pouvoir faire le lien avec les blocs de la modélisation structurelle et celle de pouvoir modéliser des flux continus.

Le diagramme d'activités est noté « act », permet de décrire le comportement d'un système sous la forme d'une suite d'actions transformant des flux de matière.

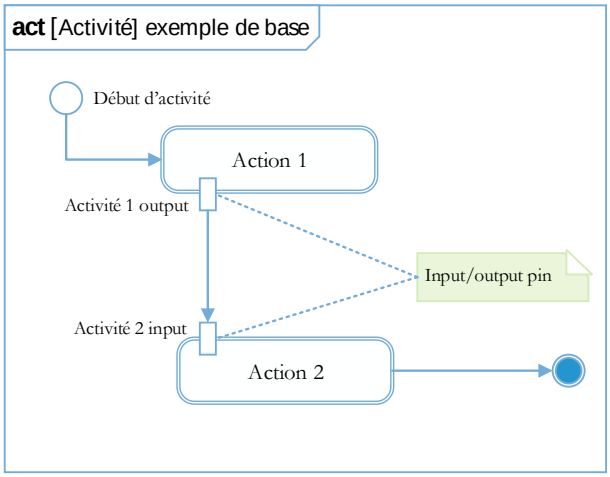
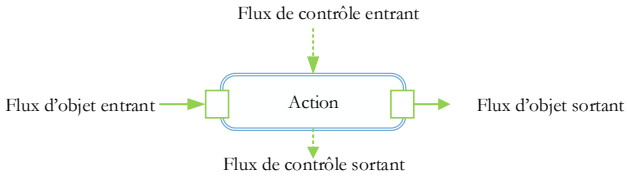
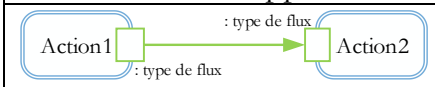
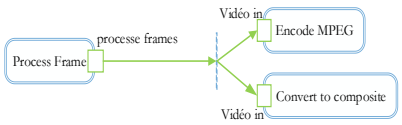
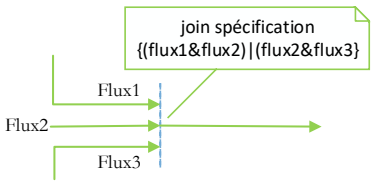


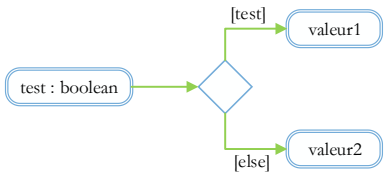
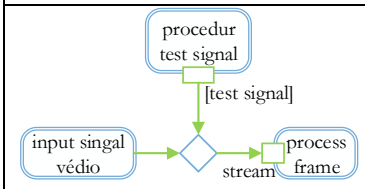
Figure I.7 : Diagramme d'activité

Une action est l'unité fondamentale de spécification comportementale en SysML. Elle représente un traitement ou une transformation. Les actions sont contenues dans les activités, qui fournissent leur contexte. Les autres éléments de base du diagramme d'activité sont les flots de contrôle entre actions, les décisions (aussi appelées branchements conditionnels), un début et une ou plusieurs fins possibles.

La notation graphique de base est l'activité en SysML démarre avec une action. Lorsque cette celui-ci est terminée, il produit un jeton qui va alimenter l'action suivant, ce jeton-là peut être un flot de contrôle ou un flux d'objet, soit d'une façon directe ou soit via l'un des nœuds proposés par SysML, toutes les activités sont limitées à deux points début et fin. La description de syntaxe de diagramme d'activité sous SysML est comme suit.

Tableau I.4 : syntaxe de relation de diagramme d'activité

	
Action : La petite unité de fonction exécutable au sein d'une activité est l'action, elle est représentée sous forme d'un rectangle arrondis, ce dernier contient le nom ou description de l'action, si une action est appel d'activité, alors un petit symbole de fourchette est apposé en bas, adroite.	
	Object Flow : Représente la participation d'un objet dans un flux de contrôle, et sont représentés par des relations de dépendance entre objets et états d'action ou d'activités.
	Control flow : Passage instantané et automatique à l'action suivante. Et déclenchées par la fin d'une activité et provoquant le début immédiat d'une autre.
	Fork : Un est un noud de contrôle structuré représentant un débranchement parallèle. Il est représenté par une barre horizontale ou verticale qui possède un arc entrant et plusieurs arcs sortants. Le fork duplique le jeton entrant sur chaque flot sortant. Les jetons sur les arcs sortants sont indépendants et concurrents
	Join : Un est un nœud de contrôle structuré représentant une synchronisation entre actions (rendez-vous). Il est représenté par une barre horizontale ou verticale qui possède un arc sortant et plusieurs arcs entrants. Le join ne produit son jeton de sortie que lorsqu'un jeton est disponible sur chaque flot entrant. Join spécification : Il s'agir d'une expression logique utilisant les noms des flots entrants, qui doivent donc être nommés pour l'occasion.
	Nœud initial : un point de départ ou la première activité de l'écoulement. (Démarrage de l'activité)
	Nœud final : L'extrémité du diagramme d'activité, également appelé comme activité finale. (Arrête d'activité)
	Nœud de fin de flux : pour montre qu'une séquence s'actions est terminées sans arrêter l'activité

	Décision nœud : Une alternative spécifie des chemins alternatifs, chacun basé sur la valeur d'une expression booléenne (condition de garde). Qui valide ou non le déclenchement d'une transition, les conditions de garde ne doivent pas se chevaucher et couvrir tous les cas possibles.
	Merge nœud : Un nœud de contrôle qui rassemble de multiple flux alternatif entrant et accepte un flux sortant unique. Merge ne doit pas être utilisé pour synchroniser les flux simultanés.

I.6 L'ingénierie dirigée par modèles

Au sens large du terme, l'Ingénierie Dirigée par les Modèles (IDM), utilisée encore sous le terme anglais de MDE (Model Driven Engineering), est la discipline qui place les modèles au centre des processus d'ingénierie système. Car peut être définie comme le traitement symbolique du contenu des modèles, par exemple par enrichissement, extension, interprétation, traduction, transformation, génération, etc. Avec cette approche, le code final exécutable n'est plus considéré comme l'élément central dans le processus de développement mais comme un élément qui résulte d'une transformation de modèles.

Il existe plusieurs approches de développement basées sur l'IDM, tel que le MIC (Model-Integrated Computing) a été développée par *Sztipanovits et al* en 1995, Model Driven Architecture (MDA) développé par *OMG* en 2001 et les usines logicielles (Software factories) de *Microsoft* en 2003. Dans ce qui suit nous présentons approche de la conception Model Driven Architecture (MDA).

I.7 Architecture dirigée par les modèles

"Le Object Management Group (OMG) avait rendu public le MDA en 2001 en tant qu'approche pour l'ingénierie dirigée par les modèles (MDE). Dans cette approche, les modèles se sont davantage concentrés sur le processus de développement et suscitent un intérêt croissant de la part des laboratoires industriels et de recherche. MDA sépare la spécification fonctionnelle et la tâche de modélisation d'un système des détails d'implémentation sur une plate-forme donnée, sans perdre l'intégration du modèle dans une plate-forme cible. La MDA considère les modèles à transformer comme un élément productif utilisé pour générer automatiquement le code source de l'application. Les principales technologies de l'approche MDA sont le langage UML (Unified Modeling Language), la fonction MOF (Meta-Object Facility), l'échange XMI (XML Metadata Interchange), Query / View / Transformation (QVT) et Acceleo". [1]

OMG a défini certains termes autour des modèles nommés méta-métamodèle, méta-modèle et modèle, et MDA définit également trois niveaux de modèles différents : modèle indépendant du calcul (CIM), modèle indépendant de la plate-forme (PIM) et modèles spécifiques de la plate-forme (PSM). Pour

modéliser l'application puis les transformations successives pour générer le code source. Ces modèles sont illustrés aux fig I.8 et fig I.9.

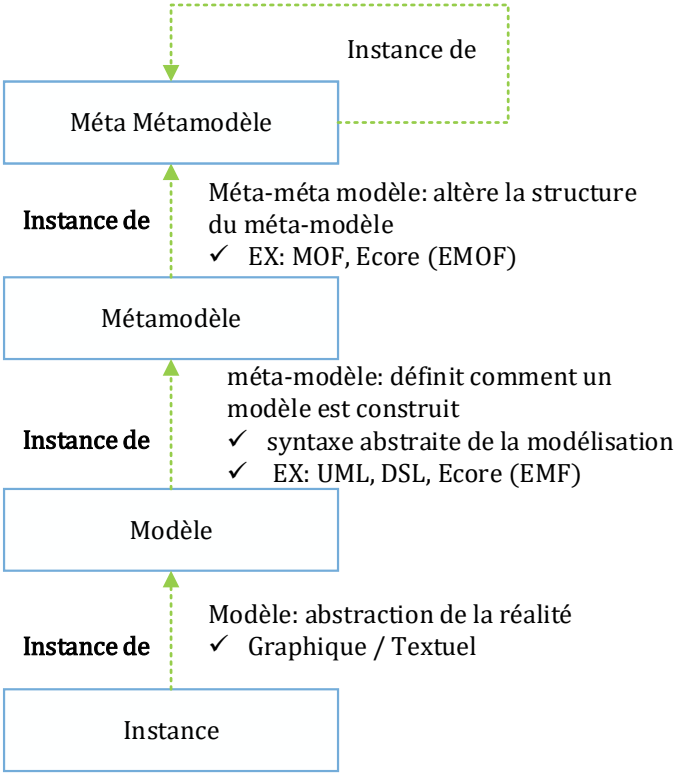


Figure I.8: Quatre niveaux d'architecture de métamodèle

Ainsi, la norme MDA produit des modèles cibles à partir d'un certain nombre de modèles sources et permet de mettre en œuvre le même modèle sur plusieurs plates-formes au moyen de projections normalisées. Il permet aux applications d'interagir avec les modèles et de soutenir le développement de nouvelles techniques et plateformes. La mise en œuvre de la MDA est entièrement basée sur les modèles et leurs transformations.

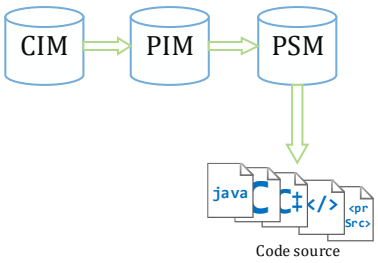


Figure I.9: Modèles clés en MDA

Les transformations entre les différents modèles sont produites avec des outils compatibles avec la norme OMG appelée QVT (Query / View / Transformation). La transformation des entités de la source du modèle comporte deux étapes. La première étape consiste à identifier les correspondances entre les concepts de modèles source et cible dans leurs méta-modèles, qui indiquent l'existence d'une fonction de transformation applicable à toutes les instances du

méta-modèle source. Ensuite, la deuxième étape consiste à appliquer la transformation du modèle source pour générer automatiquement le modèle cible à l'aide d'un programme appelé moteur de transformation ou exécution.

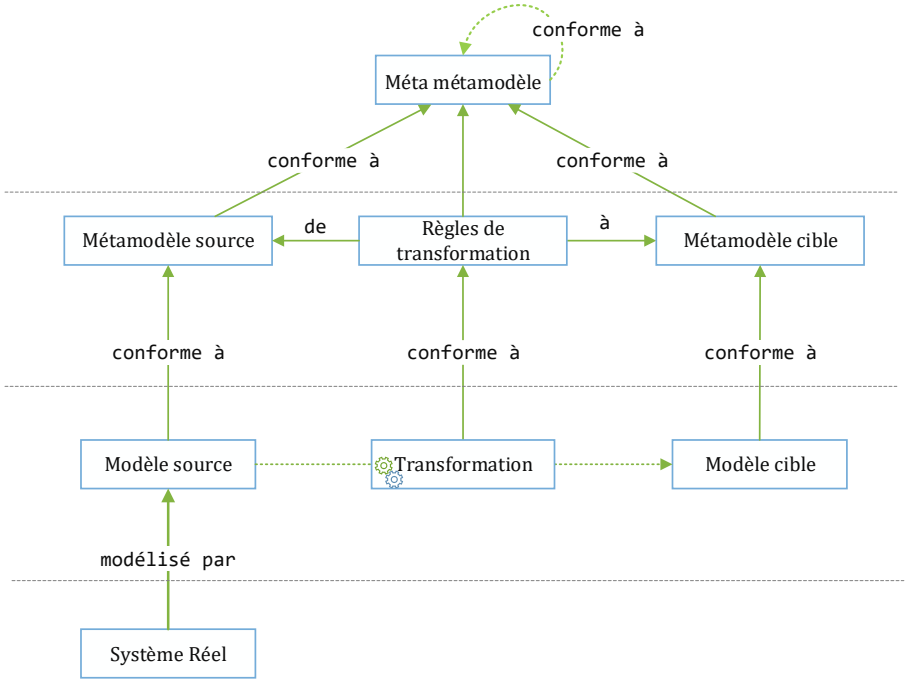


Figure I.10 : Processus de transformation dans l'approche MDA

"La transformation de modèle consiste à convertir un modèle en un autre dans le même système. La transformation produit un modèle spécifique à la plate-forme (PSM) en combinant le modèle indépendant de la plate-forme (PIM) et des informations supplémentaires. Quelle que soit la plate-forme cible ultime pouvant être la première étape lors de la construction d'applications basées sur MDA, vous devez créer un PIM exprimé via UML. Ce modèle général peut ensuite être transformé en un ou plusieurs PSM tels que .NET, EJB, etc.

Un mappage MDA fournit des spécifications sur la manière de transformer un PIM en un PSM particulier. Le modèle de plate-forme cible définit la nature du mappage. Une partie de la transformation peut résulter d'un exercice manuel, mais l'intention est clairement d'automatiser un grand nombre de processus, dans la mesure des outils utilisés.

Un mappage de transformation de modèle doit être défini à l'aide d'un langage, qu'il s'agisse d'un langage d'action, d'un langage naturel ou d'un langage de mappage dédié. Les transformations entre les différents modèles sont réalisées à l'aide d'outils compatibles avec la norme OMG appelée MOF QVT.

Un enregistrement de la transformation doit inclure un graphique indiquant quels éléments PIM ont été mappés avec quels éléments PSM et comprend les règles de mappage utilisées pour chaque partie de la transformation. La génération de code est la dernière étape du processus de transformation, de sorte que la sémantique complète de l'exécution et de l'application puisse être incluse dans le PSM".[20]

1.8 Outils MBSE

Les systèmes basés sur des modèles sont les plus robustes, car la méthodologie MBSE offre des considérables avantages, en réduisant les inefficacités, amélioré la qualité de la conception et en éliminant les erreurs qui se produisent dans le développement d'outils anciens (tels que SDLC). Il est important de mentionner qu'un outil est un cadre est un environnement complet prenant en charge un ensemble d'outils pouvant être utilisés pour effectuer diverses activités MBSE. Le *tab I.5* présentés les outils couramment utilisés pour effectuer des activités de développement de modèles et prennent en charge différents profils UML/SysML. Et fournissent également une prise en charge partielle d'autres activités MBSE. Par exemple la transformation et la vérification de modèle. Cependant, certain d'aux sont disponibles pour effectuer des transformations de modèle à modèle ou de modèle à texte.

Tableau I.5 : Outils MBSE les plus utilisés

Nom	Editeur	Environnement	Fonctionnalité	License
Enterprise Architect	Sparx Systems	UML, SysML, BPMN	Modélisation	Payante
Cameo Systems Modeler	No Magic	SysML	Modélisation	Payante
Rational Rhapsody Designer	IBM	UML, SysML ou des DSL (AUTOSAR)	Modélisation	Payante
Modelio SA	ModelioSoft	SysML, UML et BPMN, NAF/DoDAF/ MODAF	Modélisation	Payante
Capella	Projet PolarSys, OBEO	UML, SysML, BPMN	Modélisation	Open-source
Papyrus	Projet PolarSys, CEA LIST	UML, SysML, BPMN, MARTE	Modélisation	Open-source
MediniQVT	EPL	UML, SysML	Modélisation Transformation(M2M)	Open-source
ATL	EPL	UML, SysML	Modélisation Transformation (M2M)	Open-source
Acceleo	EPL	UML, SysML	Modélisation Transformation(M2T)	Open-source

Cependant, il n'existe pas d'outil parfait. Chaque projet est unique, et les outils MBSE qui ont bien fonctionné pour un projet peuvent ne pas convenir du tout à un autre. Un engagement envers la qualité, associé à la flexibilité, est nécessaire pour garantir que les systèmes basés sur des modèles sont construits correctement à chaque fois. Dans notre travail on a choisi l'environnement Polarsys, qui prend en charge la solution Papyrus pour la modélisation SysML, et le Acceleo pour la transformation des modèles.

I.9 Conclusion

Dans ce chapitre, nous avons expliquée brièvement l'ingénierie systèmes, ainsi que l'application formalisée de la conception basée sur les modèles et le langage de modélisation SysML. La présentation graphique des diagrammes SysML garantit la facilité et la flexibilité en tant que support de conception multidisciplinaire, bien qu'elle souffre principalement d'une sémantique formelle faible. L'IDM offre la possibilité d'automatiser le passage entre des langages de modélisation. Représenter les connaissances introduites dans une modélisation formellement peut servir lors de l'analyse de cette dernière. Le Web sémantique offre des langages facilitant formalisation et le partage des connaissances et leur réutilisation. Dans le chapitre suivant, nous présenterons le web sémantique et ses langages associés.

Chapitre II

Représentation des Connaissances et Raisonnement

II.1 Introduction

Le développement du Web vers Web 3.0 qu'est connu sous le nom le web sémantique a renouvelé le domaine de web en introduisant la notion des ontologies. Qui sont souvent capturées dans des langages de représentation de la connaissance (KR :Knowledge Representation) qui proviennent des théories de l'intelligence artificielle. Ces langages sont pour la plupart structurés suivant des formalismes de logique tels que la logique de prédicats. D'autre part, le Web sémantique utilisé pour désigner les formats et les technologies qui le permettent. La collecte, la structuration et la récupération des données liées sont activées par des technologies fournissant une description formelle des concepts, des termes et des relations dans un domaine de connaissances donné. Ces technologies sont spécifiées en tant que normes W3C.

II.2 Le Web sémantique

"Le Web sémantique désigne un ensemble de technologies visant à rendre le contenu des ressources du World Wide Web accessible et utilisable par les programmes et agents logiciels, grâce à un système de métadonnées formelles, utilisant notamment la famille de langages développés par le W3C". [18]

Le Web sémantique étend le réseau des hyperliens entre des pages Web classiques par un réseau de lien entre données structurées permettant d'effectuer des tâches recherche, apprentissage, etc. plus précises pour les utilisateurs.

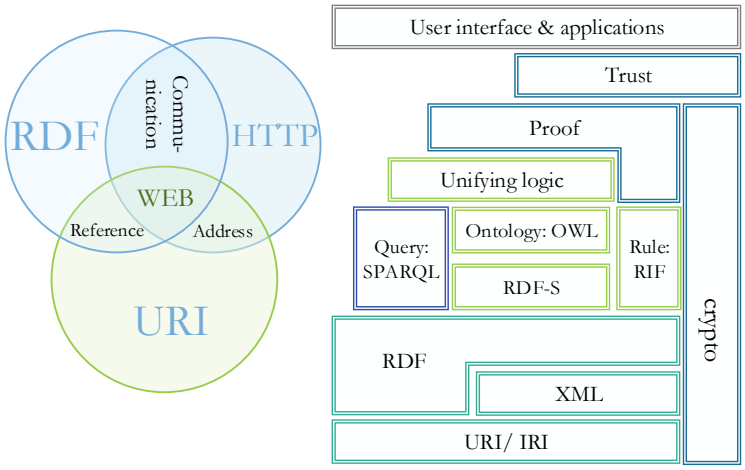


Figure II.1 : différentes briques technologiques du Web sémantique

Les différentes technologies qui se cachent derrière ce terme de Web sémantique on peut les regroupe comme suit :

- Représentation

  URI/IRI : Universal Ressource Identifier/International Ressource Identifier

  XML : Extensible Markup Language

  RDF : Ressource Description Framework : description des ressources sous forme de graphe à base de triplets

- Raisonnement

  RDFS : RDF Schema : langage de description de vocabulaire associé à RDF (description de classes et propriétés)

  OWL : Ontology Web language : langage de représentation des ontologies

  RIF : Rule Interchange Format : échange de systèmes à base de règles

  SWRL : Semantic Web Rule Language. il inclut les règles de haut niveau car ils sont exprimées en termes de concepts OWL.

- Interrogation

  SPARQL : Simple Protocol And Rdf Query Language : language d'interrogation de graphe RDF

II.3 Description des ressources RDF

II.3.1 URI

"Uniform Resource Identifier est une courte chaîne de caractères identifiant une ressource sur un réseau (par exemple une ressource Web) physique ou abstraite, et dont la syntaxe respecte une norme d'Internet mise en place pour le World Wide Web. " [16]

Scheme:	[//authority/]	[path]	[?query]	[#fragment]

Figure II.2 : URI syntaxe générique

- **scheme:** http,ftp,mailto,…
- **authority:** [userinfo@]host[:port],
  - userinfo: ex.: username: password
  - host: domaine-nom,)adresse IP
  - port: numéro de port, ex: 80 le port HTTP standard
- **path:** séquence de segments (similaires à des répertoires), séparés par des /, ex. : chemin dans un le système de fichiers d'un serveur HTTP
- **query:** contient des informations complémentaires qui ne sont pas de nature hiérarchique (ex: suite de paires <cle>=<valeur>)séparées par & pour requêtes HTTP
- **fragment:** fournit des informations supplémentaires permettant d'accéder à une ressource secondaire(ex:id d'une ancre pour document HTML)

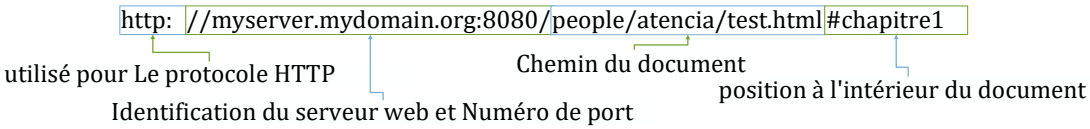


Figure II.3: Exemple de URI

"Un URI peut être de type « locator » ou « name » ou les deux.

Un Uniform Resource Locator (URL) est souvent utilisé comme synonyme de l'expression « adresse web », qui désigne une chaîne de caractères utilisée pour identifier les ressources du World Wide Web document HTML, image, son, ...etc. selon les protocoles internet http et https. Mais plus fournit les moyens d'agir sur une ressource ou d'obtenir une représentation de la ressource en décrivant son mode d'accès primaire ou « emplacement » réseau. " [8] Par exemple, l'URL http://www.univ-medea.dz/ est un URI qui identifie une ressource (page d'accueil université de Médéa) et implique qu'une représentation de cette ressource (une page HTML en caractères encodés) peut être obtenue via le protocole HTTP depuis un réseau hôte appelé www.univ-medea.dz.

"Un Uniform Resource Name (URN) est un URI qui identifie une ressource par son nom dans un espace de noms. Un URN peut être employé pour parler d'une ressource sans que cela préjuge de son emplacement ou de la manière de la référencer. " [8] Par exemple, l'URN *urn:isbn:0-395-36341-1* est un URI qui, étant un numéro de l'International Standard Book Number (ISBN), permet de faire référence à un livre, mais ne suggère ni où, ni comment en obtenir une copie réel

II.3.2 XML

"Extensible Markup Language est un métalangage informatique de balisage générique. Sa syntaxe est dite « extensible » car elle permet de définir différents langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS, SVG… Elle est reconnaissable par son usage des chevrons (<, >) encadrant les noms des balises. Il est utilisé pour le stockage de documents que pour la transmission de données entre applications. Sa simplicité, sa flexibilité et ses possibilités d'extension ont permis de l'adapter à de multiples domaines allant des données géographiques au dessin vectoriel en passant par les échanges commerciaux. " [16] De nombreuses technologies se sont développées autour de XML et enrichissent ainsi son environnement.

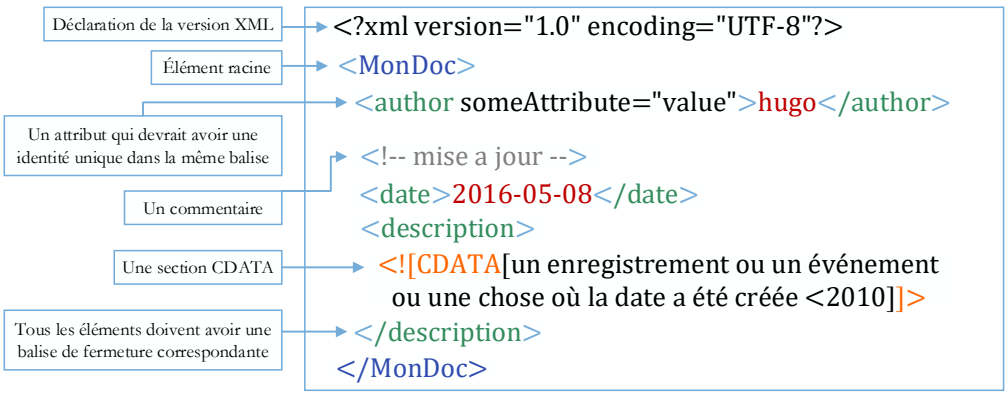


Figure II.4 : Exemple de document XML

Pour garantir l'unicité des noms des éléments qui les utilisent dans un document XML, le W3C a fourni dans une recommandation la notion d'espaces de nommage XML *namespaces*. Une instance XML peut contenir des noms d'élément ou d'attribut provenant de plus d'un vocabulaire XML. Si chaque vocabulaire se voit attribuer un espace de noms, l'ambiguïté entre les éléments ou attributs du même nom peut être résolue.

Ce qui permettent à XML de réutiliser des éléments / attributs d'autres documents XML au lieu de les recréer à chaque fois. Une instance XML peut contenir des noms d'élément ou d'attribut provenant de plusieurs documents XML. Il sera impossible de conserver des éléments de différents documents uniques, en particulier lors de l'utilisation du document de quelqu'un d'autre. L'espace de noms est un moyen de résoudre les conflits de noms entre différentes sources d'élément (par exemple, les sources XSD).

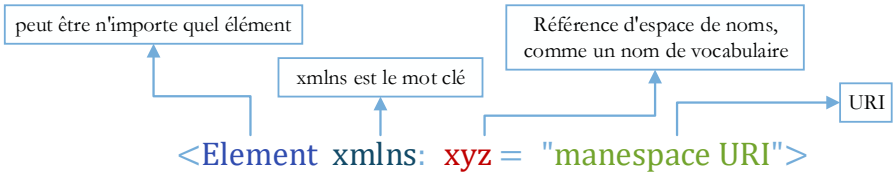


Figure II.5 : Exemple de syntaxe de XML namespaces

II.3.3 RDF

"Resource Description Framework (RDF) est un modèle de graphe destiné à décrire de façon formelle les ressources Web et leurs métadonnées, de façon à permettre le traitement automatique de telles descriptions. Développé par le W3C, RDF est le langage de base du Web sémantique. L'une des syntaxes (ou sérialisations) de ce langage est RDF/XML. D'autres sérialisations de RDF sont apparues ensuite, cherchant à rendre la lecture plus compréhensible ; c'est le cas par exemple de Notation3 (ou N3). " [31]

En annotant des documents non structurés et en servant d'interface pour des applications et des documents structurés RDF permet une certaine interopérabilité entre des applications échangeant de l'information non formalisée et non structurée sur le Web.

L'atome de connaissance en RDF est le triplet (Sujet, Prédicat, Objet) Représentation sous forme de graphe

— Sujet (Ressource) est une entité d'informations pouvant être référencée par un identificateur. Cet identificateur doit être une URI. Les sujet 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…etc.

— Prédicat (Propriété) est un aspect spécifique, un attribut, une caractéristique ou relation utilisée pour décrire une ressource.

— Objet est une valeur de la propriété peut être une ressource ou littéral (simple chaîne de caractères)

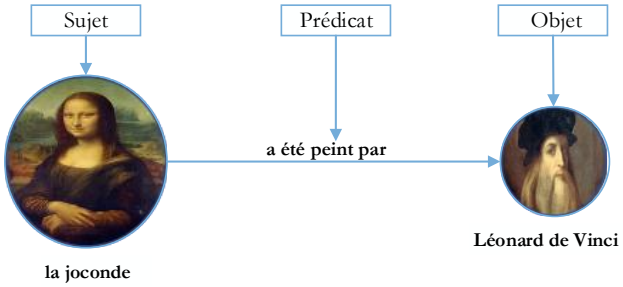


Figure II.6 : Exemple de triple RDF

Le Graphe RDF est un modèle abstrait bien adapter à une interprétation humaine, la sérialisation RDF fournir un moyen de conversion entre le modèle abstrait et un format concret plusieurs formats de sérialisation sont apparues d'abord RDF / XML, les syntaxes qui sont apparues ensuite cherchant à rendre la lecture plus compréhensible. C'est le cas par exemple de Notation3(N3). Dans l'exemple suivant, nous allons montrer les différentes présentations RDF/XML, N-Triples et Turtle.

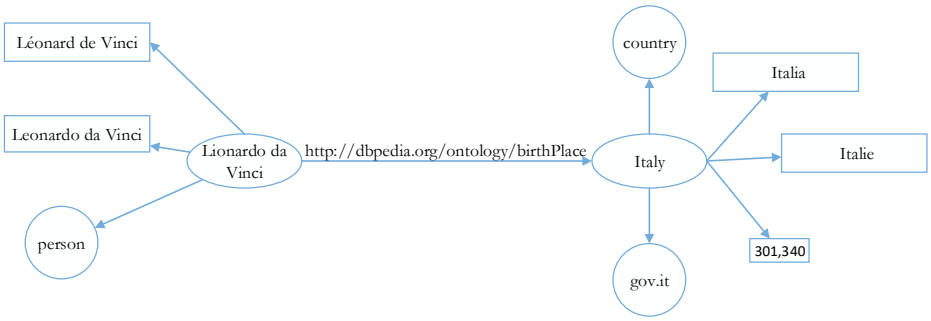


Figure II.7 : Représentation graphique de RDF

RDF/XML

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:foaf="http://www.xmlns.com/foaf/1.0"
xmlns:drfs="http://www.w3.ogr/2000/01/rdf-schema#"
xmlns:ns0="http://dbpedia.org/ontology/"
xmlns:schema="http://scheam.org"
xmlns:geo="http://www.w3.ogr/2003/01/geo/wgs84_pos#">

 <foaf:Person rdf:about="http://dbpedia.org/resource/Lionardo_da_Vinci">
  <rfds:label xml: lang="it">Leonardo da Vinci </rfds:label>
  <rfds:label xml: lang="fr">Léonard de Vinci </rfds:label>
  <ns0:birthPlace>
   <schma:Country rdf:about="http://dbpedia.org/resource/Italy">
    <rfds:label xml: lang="it">Italia </rfds:label>
    <rfds:label xml: lang="fr">Italie </rfds:label>
    <geo:lat rdf:datatype="http://w3.org/2001/XMLSchema#float">301,340</geo:lat>
    <foaf:homepage rdf:resource="http://gov.it/"/>
   </scheam:Country>
  </ns0:birthPlace>
 </foaf:Person>
</rdf:RDF>
```

Figure II.8 : Représentation XML de RDF

N-Triples

```
<http://dbpedia.org/resource/Lionardo_da_Vinci><http://www.w3.org/1999/02/22-rdf-syntax-ns#type><http://xmlns.com/foaf/1.0/Person>
<http://dbpedia.org/resource/Lionardo_da_Vinci><http://www.w3.ogr/2000/01/rdf-schema#label>"Leonardo da Vinci"@it
<http://dbpedia.org/resource/Lionardo_da_Vinci><http://www.w3.ogr/2000/01/rdf-schema#label>"Léonard de Vinci"@fr
<http://dbpedia.org/resource/Lionardo_da_Vinci><http://dbpedia.org/ontology/birthPlace><http://dbpedia.org/resource/Italy>
<http://dbpedia.org/resource/Italy><http://www.w3.org/1999/02/22-rdf-syntax-ns#type><http://schema.org/Country>
<http://dbpedia.org/resource/Italy><http://www.w3.ogr/2000/01/rdf-schema#label>"Italia"@it
<http://dbpedia.org/resource/Italy><http://www.w3.ogr/2000/01/rdf-schema#label>"Italie"@fr
<http://dbpedia.org/resource/Italy><http://www.w3.org/2003/01/geo/wgs84_pos#lat>301.340^^<http://www.w3org/2001/XMLSchema#foaf>
<http://dbpedia.org/resource/Italy><http://xmlns.com/foaf/1.0/homepage><http://gov.it>
```

Figure II.9 : Représentation N-triples de RDF

Turtle

```
@prefix bdr: <http://dbpedia.org/resource/>
@prefix dbo: <http://dbpedia.org/ontology/>
@prefix rdfs: <hhtp://www.w3.ogr/2000/01/rdf-schema#>
@prefix foaf: <http://xmlns.com/foaf/1.0/>
@perfix geo: <http://www.w3.ogr/2003/01/geo/wgs84_pos#>
@perfix xsd: <http://www.w3.org/2001/XMLSchema#>
@perfix schema: <http://schema.org/>

dbr:Lionardo_da_Vinci
 a foaf: Person;
 rdfs:label"Leonardo da Vinci"@it;
 rdfs:label"Léonard de Vinci"@fr;
 bdo:birthplace dbr:Italy.

dbr:Italy
 a schema:Country;
 rdfs:label"Italia"@it;
 rdfs:label"Italie"@fr;
 geo:lat "301.340"^^xsd:float;
 foaf:haomepage<http://gov.it>.
```

Figure II.10 : Représentation turtle de RDF

II.4 Description des ressources RDF-S

RDF Schema ou RDF-S est un langage extensible de représentation des connaissances. Il appartient à la famille des langages du Web sémantique publiés par le W3C. RDF-S fournit des éléments de bases pour la définition d'ontologies ou vocabulaires destinés à structurer des ressources RDF. Avec ces ressources structurées avec RDF-S dans un triplestore, vous pouvez utiliser le langage de requête SPARQL pour les atteindre à travers le Web.

La première version de RDF-S a été proposée en mars 1999, et la recommandation finale publiée par le W3C en février 2004. Les composants principaux de RDF-S sont intégrés dans un langage d'ontologie plus expressif, OWL. [16]

"RDF-S est un langage de description de vocabulaire offre aux communautés d'utilisateurs la possibilité de définir des vocabulaires (ensembles de termes) qu'ils pourront utiliser ultérieurement dans des instructions RDF. " [30]

Un vocabulaire RDF est une instruction de classe, instructions de propriété et relations entre elles. Parfois appelés ontologies légères.

Tout comme les langages de programmation orientés objet (ex : java) RDF-S fournit un système de types pour RDF où :

— Classes : types
— Attributs : propriétés
— Instances : ressources

— Héritage : organisation hiérarchique des types

Mais RDF-S est un langage de description, pas un langage de programmation (pas de méthode)

La classe A est un ensemble des objets (individus) partage certain caractéristiques. C'est n'importe quelle déclaration de type **`rdfs: Class`** utilise la propriété **`rdf : type`**.

```
ex:MotorVehicle rdf:type rdfs:Class
```

La ressource **`rdfs: Class`** fait référence à la classe de toutes les classes RDF. La relation de spécialisation entre deux classes est décrite à l'aide de la propriété prédéfinie **`rdfs: subClassOf`**.

```
ex:MotorVehicle  rdf:type rdfs:Class.
ex:Van             rdf:type rdfs:Class;
                  rdfs:subClassOf ex:MotorVehicle.
```

Tableau II.1 : Les classes RDF/RDF-S

Classe	Commentaire
rdfs:Resource	Tout est ressource
rdfs:Literal	Un littéral donc du texte
rdf:langString	Un littéral avec une indication de langue (par exemple @fr)
rdf:HTML	Littéral HTML
rdf:XMLLiteral	Littéral XML
rdfs:Class	Une classe
rdf:Property	Une propriété
rdfs:Datatype	Un type de données
rdf:Statement	Un énoncé RDF (s,p,o)
rdf:Bag	Un container non ordonné
rdf:Seq	Un container ordonné
rdf:Alt	Un container d'alternatives
rdfs:Container	Un container RDF
rdfs:ContainerMembershipProperty	Une propriété d'appartenance à un container (rdf:_1, rdf:_2, … ssTprop de rdfs:member)
rdf:List	Une liste RDF (rdf:nil est instance de rdf:List)

Les propriétés sont décrites comme des instances de la classe **`rdf: Property`**. Cette classe indique la classe de toutes les propriétés. Les propriétés RDF-S **`rdfs: domain`**, **`rdfs: range`** et **`rdfs: subPropertyOf`** décrivent comment les propriétés et les classes sont destinées à être utilisées ensemble dans les données RDF.

`rdfs: domain` spécifie le type de tous les individus qui font l'objet d'instructions à l'aide de la propriété décrite.

`rdfs: range` spécifie le type de tous les individus ou le type de données de tous les littéraux qui font l'objet d'instructions utilisant la propriété décrite.

```
ex:Person   rdf:type      rdfs:Class.
ex:Book     rdf:type      rdfs:Class.
ex:author   rdf:type      rdf:Property;
            rdfs:domain   ex:Book;
            rdfs:range    ex:Person.
```

Tabeau II.2 : Les propriétés RDF/RDF-S

Propriété	Domaine	Co/domaine
rdf:type	rdfs:Resource	rdfs:Class
rdfs:subclassOf	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	rdf:Property	rdf:Property
rdfs:domain	rdf:Property	rdfs:Class
rdfs:range	rdf:Property	rdfs:Class
rdfs:label	rdfs:Resource	rdfs:Literal
rdfs:comment	rdfs:Resource	rdfs:Literal
rdfs:member	rdfs:Resource	rdfs:Resource
rdf:first	rdf:List	rdfs:Resource
rdf:rest	rdf:List	rdf:List
rdfs:seeAlso	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	rdfs:Resource	rdfs:Resource
rdf:value	rdfs:Resource	rdfs:Resource
rdf:subject	rdfs:Statement	rdfs:Resource
rdf:predicate	rdfs:Statement	rdfs:Resource
Rdf:object	rdfs:Statement	rdfs:Resource

II.5 Représentation de données OWL

"Le langage d'ontologie Web (OWL) existe une deuxième version OWL2 depuis 2012. Il est plus expressif que XML, RDF ou RDFS en fournissant un vocabulaire supplémentaire ainsi qu'une sémantique formelle car Les ontologies OWL 2 intègrent des classes, des propriétés, des individus et des valeurs de données et sont stockées dans des documents Web Sémantique. Les ontologies OWL 2 peuvent être utilisées de concert avec des informations décrites à l'aide de RDF et les ontologies OWL 2 sont elles-mêmes principalement échangées sous la forme de documents RDF.

OWL2 existe dans différentes sous-langues : OWL Lite, OWL DL et OWL Full. Une caractéristique importante de chaque sous-langage est son expressivité. OWL Lite est le moins expressif et le OWL Full est le sous-langage le plus expressif. Autre spécification de OWL Lite dans la deuxième version de OWL on distinguons trois profils distinctes OWL2 QL, OWL2 RL et OWL2 EL. " [17]

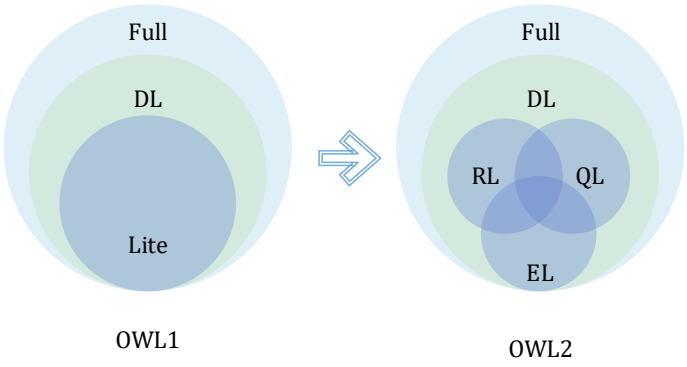


Figure II.11: Sous-langages OWL

"OWL Full est le plus expressif des sous-langages OWL et utilise l'ensemble des primitives du langage OWL. Il est destiné à être utilisé dans des situations où une très grande expressivité est plus importante que la garantie de décidabilité ou de complétude informatique du langage. Ce sous-langage est destiné aux utilisateurs qui veulent un maximum d'expressivité et la liberté syntaxique de RDF, mais sans aucune garantie informatique.

OWL DL est une sous-langue de OWL Full qui limite l'application des constructeurs OWL et RDF. OWL2 DL (logique de description) n'est pas compatible avec RDF, de la même manière que tous les documents RDF ne sont pas des documents légaux OWL DL, bien que tous les documents légaux OWL2 DL soient des documents RDF légaux. Cette sous-langue prend en charge les utilisateurs qui souhaitent une expressivité maximale sans perdre la complétude et la décidabilité informatiques. " [17]

"OWL2 EL est conçu comme un sous-ensemble de OWL2 DL particulièrement adapté aux applications utilisant des ontologies définissant un très grand nombre de classes et / ou de propriétés, saisissant le pouvoir expressif utilisé par de nombreuses ontologies de ce type et pour lesquelles la cohérence d'ontologies, l'expression de classe subsomption, et la vérification d'instance peut être décidée efficacement (en Temps polynomial).

OWL2 QL contient l'intersection de RDFS et OWL2 DL. Il est conçu pour que les données stockées dans un système de base de données relationnelle standard puissent être interrogées via une ontologie via un mécanisme de réécriture simple, c'est-à-dire en réécrivant la requête dans une requête SQL à laquelle le système SGBDR répond ensuite, sans aucune modification apportée. Les données. OWL2 QL est basé sur la famille de logiques de description DL-Lite et a été conçu pour que la réponse complète et rationnelle aux requêtes soit efficace en termes de calcul.

OWL2 RL est destiné aux applications qui nécessitent un raisonnement évolutif sans sacrifier trop de pouvoir expressif. Il est conçu pour prendre en charge à la fois les applications OWL2 DL capables d'exploiter toute l'expressivité du langage en termes d'efficacité et les applications RDF-S qui nécessitent une plus grande expressivité de la part d'OWL2. La conception de OWL2 RL a été inspirée par Description Logic Les programmes et leur expressivité ont été obtenus en définissant un sous-ensemble syntaxique de OWL2, qui peut être mis en œuvre à l'aide de technologies basées sur des règles. " [40]

II.5.1 La base de connaissances en logique descriptive

"La formalisation des connaissances pour le représenter dans ce domaine d'application permettent de représenter des concepts (aussi appelés classes) d'un domaine et les relations (aussi appelées rôles) qui peuvent être établies entre les instances de ces classes. Par exemple, on pourrait les utiliser pour représenter le concept humain, femme et mère, et spécifier que toute mère est une femme qui est le parent d'au moins un humain. Comme toute logique, des mécanismes d'inférence y sont associés, permettant ainsi de déduire de nouveaux faits à partir d'une base de connaissances.

On distingue deux composantes : la TBox et la ABox. La première contient tous les axiomes définissant les concepts du domaine, comme la définition du concept de mère présenté au paragraphe précédent. La ABox contient des assertions sur des individus, en spécifiant leur classe et leurs attributs. C'est dans la ABox qu'on indiquerait que Marie est une femme et qu'elle a deux enfants. Le type d'inférence réalisé avec la TBox diffère de celui réalisé avec la ABox. " [22]

Dans la TBox, on est généralement intéressé à savoir si tous les concepts définis sont consistants, c'est-à-dire si, pour chaque concept, il peut exister au moins un individu membre de cette classe. Par exemple, si on définit une classe comme étant à la fois une sous-classe des classes homme et femme et que la TBox spécifie aussi que ces deux classes sont disjointes (c'est-à-dire qu'aucune entité ne peut à la fois être un homme et une femme), on se retrouve alors avec un concept inconsistant. Un autre type d'inférence réalisé avec la TBox est la subsomption, qui consiste à déduire qu'une classe est une sous-classe d'une autre classe, même si cela n'est pas déclaré explicitement dans la base de connaissances. Par exemple si on spécifie qu'humain est une sous classe d'animal, on peut déduire qu'une mère est un animal.

Dans la ABox, on retrouve les assertions sur les individus. En d'autres mots, on y spécifie quelles sont les entités du monde et à quelle classe elles appartiennent. C'est dans la ABox, par exemple, qu'on spécifiera que Marie est une mère, c'est-à-dire un individu qui est une instance de la classe mère. La ABox contient aussi des énoncés spécifiant les relations qui existent entre les individus. Ainsi, comme dans la TBox il sera spécifié qu'une mère doit avoir au moins un enfant, la ABox devra contenir au moins un autre individu, et une relation entre celui-ci et Marie indiquant qu'il est un de ses enfants.

Les inférences avec la ABox visent normalement à déterminer si un ensemble d'assertions est consistant, c'est-à-dire si un individu déclaré comme instance d'une classe peut réellement être une instance de cette classe, et similairement, si une relation déclarée entre deux individus est réellement possible. Supposons par exemple qu'une TBox déclare qu'un célibataire est une personne non mariée. La ABox sera inconsistante si elle contient un célibataire qui est marié avec une autre personne.

II.5.2 Syntaxe et sémantique OWL

Comme nous l'avons déjà expliqué, les ontologies écrites dans OWL sont basées sur la logique de description ou la logique de premier ordre (FOL). En effet, les différentes versions de OWL ne sont que des variantes syntaxiques de fragments de FOL où objets, ensembles (ou classes) d'objets et leurs relations peuvent être décrits formellement avec les contraintes correspondantes.

- **Élément de classe**

Les classes sont des collections d'individus. Ils sont définis dans un document OWL avec l'élément **owl:Class**. Par exemple, la classe « Enseignant » peut être définie comme suit

```
<owl:Class rdf :ID= ”Enseignant”>
     <rdfs:subClassOf Rdf:resource=”#Person”/>
</owl:Class>
```

L'élément **rdf: ID** définit le nom de la classe. Pour faire référence à une classe, en utilisent l'élément **rdf: resource**. Une ontologie OWL peut représenter la hiérarchie entre les classes à l'aide de l'élément **owl:subClassOf**. Par exemple, la classe «Enseignant» est une sous-classe de «Person». La sémantique de cette affirmation est $\forall x$ Enseignant (x) $\Rightarrow$ Personne (x) .

Il est possible d'établir des relations entre deux classes, en utilisant des éléments **owl:equivalentClass** et **owl:disjointWith**. L'assertion **owl:equivalentClass** appliquée aux classes A et B représente que la classe A à les mêmes individus que la classe B. Par exemple, la classe "faculté" est équivalente à la classe "MemberStaffAcadémique":

```
<owl :class rdf :ID= “faculté”>
 <owl:equivalentClass rdf:resource=“# MemberStaffAcadémique”/>
</owl:Class>
```

L'élément **owl: disjointWith** appliqué à deux classes A et B spécifie que les classes A et B sont disjointes, c'est-à-dire que les classes A et B n'ont aucun membre en commun. Par exemple, un « professeur titulaire» ne peut pas être un «professeur associé» en même temps.

```
<owl:Class rdf:about=“#ProfesseurAssocié”>
   <owl:disijointWith rdf:resource=“#ProfesseurTitulaire”/>
</owl:class>
```

Classes prédéfinies en OWL sont les classe **owl :Thing** et **owl:Nothing** où **owl:Thing** est la classe de tous les individus c'est-à-dire toutes les classes OWL sont des sous-classes de **owl:Thing**, par contre **owl:Nothing** est la classe dont l'extension est vide cela signifie que **owl:Nothing** est sous-classe de toutes les classes OWL.

- **Classe complexe**

Une autre façon de créer des classes (plus complexes) dans OWL consiste à combiner des classes simples à l'aide d'opérateurs booléens (union, intersection et complément). L'élément **owl:unionOf** appliqué aux classes A et B crée une nouvelle classe contenant tous les membres des classes A et B. Par exemple, la combinaison de la classe « membre du personnel » et de la classe « étudiant » crée la nouvelle classe « PersonnesUniv ».

```
<owl:Class rdf :ID=" PersonnesUniv">
   <owl:unionOf rdf :parseType="Collection">
       <owl:Class rdf:about="#MembrePersonnel"/>
       <owl:Class rdf:about="#Etudiant "/>
   </owl:unionOf>
<owl:Class>
```

L'élément **owl:intersectionOf** crée une nouvelle classe à partir des deux classes A et B, qui comporte des éléments appartenant à la fois aux classes A et B.

```
   <owl:Class rdf :about="faculté"/>
      <owl :inetrsectionOf rdf :parseType="Collection">
         <owl:Restriction>
             <owl:onProperty rdf:resource="Appartient_a"/>
             <owl:hasValue rdf:resource="MF_Departement"/>
         </owl:Restriction>
      </owl :inetrsectionOf>
   </owl:Class>
```

Les individus de la nouvelle classe créée ci-dessus sont ceux qui appartiennent à la fois aux classes « Faculté » et à la classe anonyme créée par la restriction de la propriété « Appartient à ». L'élément **owl:complement** sélectionne tous les individus du domaine n'appartenant pas à une certaine classe.

```
   <owl :Class rdf :about="#cours">
      <rdfs:subClassOf>
         <owl:Class>
           <owl:complementOf rdf:resource="#staffMembre"/>
         </owl:Class>
      </rdfs:subClassOf>
   </owl:Class>
```

La classe "cours" ci-dessus a pour membres tous les individus qui n'appartiennent pas à la classe « staff Membre ».

- **Propriété**

Les propriétés décrivent une sorte de relation entre les membres des classes. Dans un document OWL, deux types de propriétés sont distingués :

— Les propriétés d'objet, qui associent des objets à d'autres objets, c'est-à-dire des instances d'une classe avec des instances d'une autre classe. Par exemple, la propriété d'objet « EstEnseignéPar » relie la classe « cours » à la classe « MemberStaffAcadémique ».

```
<owl:ObjectProperty rdf :ID="EstEnseignéPar">
   <owl:domain rdf:resource="#cours"/>
   <owl:rang rdf:resource="MemberStaffAcadémique"/>
</owl:ObjectProperty>
```

— Propriétés de type de données, qui associent des objets à des valeurs de type de données. OWL n'a pas de types de données prédéfinis, mais il permet d'utiliser les types de données XMLSchema. Par exemple, l'année de naissance d'une personne est spécifiée à l'aide du type de données «http://www.w3.org/2001/XLMSchema#nonNegativeInteger» du schéma XML.

```
<owl :DatatypeProperty rdf :ID="age">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/
                XLMSchema#nonNegativeInteger"/>
<owl :DatatypeProperty>
```

- **Restrictions de propriété**

Des limites plus élaborées peuvent être établies en appliquant des restrictions à une propriété, ce qui donne les sous-catégories d'individus qui remplissent cette condition. Il existe deux types de restrictions : les contraintes de valeur et les contraintes de cardinalité. Les exemples de contraintes de valeurs incluent **owl:allValuesFrom**, **owl:someValuesFrom** et **owl:hasValues**.

owl:allValuesFrom- définit l'ensemble des individus pour lesquels toutes les valeurs de la propriété restreinte sont des instances d'une certaine classe.

```
    <owl:Class rdf :about="#CoursPierAnnée">
       <rdfs:subClassOf>
         <wol:Restriction>
           <owl:onProperty
                rdf:resource="#EstEnseignéPar"/>
           <owl:allValuesFrom
                rdf:resource="#Professeur"/>
         </wol:Restriction>
       </rdfs:subClassOf>
    </owl:Class>
```

Dans ce cas-là, les individus membres de la classe « CoursPierAnnée » sont tous les cours pour lesquels la propriété « EstEnseignéPar » est attribuée à un « professeur » someValuesFrom définit l'ensemble des individus ayant au moins une relation avec une instance d'une certaine classe.

```
<owl:Class rdf:about="#MemberStaffAcadémique">
   <rdfs:subClassOf>
     <owl:Restriction>
        <owl:onProperty rdf:resource="Enseignant"/>
        <owl:someValuesFrom
             rdf:resource="#CoursPremierCycle"/>
     </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

owl: hasValues - définit un ensemble d'individus pour lesquels la valeur de la propriété restreinte est égale à une instance donnée. Par exemple, les individus de la classe « mathCours » peuvent être caractérisés comme ceux enseignés par le professeur « 949352: »

```
   <owl:Class rdf :about="#mathCours" >
      <rdfs:subClassOf>
        <owl:Restriction>
           <owl:onProperty rdf:resource="#EstEnseignéPar">
           <owl:hasValeus rdf:resource="#949352"/>
        </owl:Restriction>
      </rdfs:subClassOf>
   </owl:Class >
```

Les contraintes de cardinalité indiquent combien de fois la propriété peut être utilisée sur une instance. Les exemples incluent **owl:maxCardinality**, **owl:minCardinality** et **owl:cardinality**.

owl:maxCardinality- définit l'ensemble des individus qui ont au plus N valeurs distinctes de la propriété concernée. Par exemple, nous pouvons spécifier que la classe « département » compte au plus 30membres

```
   <owl:Class rdf :about="#departement">
      <rdfs:subClassOf>
        <owl:Resrtriction>
           <owl:onProperty rdf:resource="#EstMembre"/>
           <owl:maxCardinality
                rdf:datatype="&xsd;nonNegativeInteger">30
        </owl:maxCardinality>
      </owl:Resrtriction>
   </rdfs:subClassOf>
</owl:Class>
```

owl:minCardinality - définit l'ensemble des individus ayant au moins N valeurs distinctes du bien concerné. Par exemple, un cours doit être enseigné par au moins un enseignant.

```
<owl:Class rdf :about="#cours">
   <rdfs:subClassOf>
     <owl:Resrtriction>
       <owl:onProperty
            rdf:resource="#EstEnseignéPar"/>
       <owl:minCardinality
            rdf:datatype="&xsd;nonNegativeInteger">1
       </owl:minCardinality>
     </owl:Resrtriction>
   </rdfs:subClassOf>
</owl:Class>
```

owl:cardinality - définit l'ensemble des individus possédant un nombre exact de valeurs distinctes du bien concerné. Cet élément est utilisé pour spécifier un nombre précis, c'est-à-dire pour exprimer le fait qu'une propriété a une cardinalité minimale égale à la cardinalité maximale.

Les propriétés ont des caractéristiques qui peuvent être défini directement pour ajouter plus d'expressivité à OWL

owl:equivalentProperty énonce une propriété qui relie les mêmes paires sujet / objet à une autre propriété.

owl: inverseOf utilisé pour définir une relation inverse entre les propriétés. Si la propriété Q est l'inverse de la propriété P, alors X est lié à Y par P, si Y est lié à X par exemple, « enseignant enseigne un cours » est l'inverse de « un cours est enseigné par un enseignant ».

owl: functionPropertyfunctional a au plus une valeur pour chaque instance.

wol: InverseFuncionalProperty sont des propriétés dont la propriété inverse est fonctionnelle. Par exemple, une propriété telle que « #EstMèreDe » est fonctionnelle inverse.

La propriété transitive est comprise comme suit : si la paire (x, y) est une instance de la propriété transitive P, et que la paire (y, z) est une instance de P, on peut en déduire la paire (x, z) est aussi une instance de P.

owl: SymmetricProperty est interprété comme suit: la paire (x, y) est une instance de A si et seulement si la paire (y, x) est également une instance de A.

II.6 Définition SPARQL

Les bases de données relationnelles traditionnelles ne sont pas la plate-forme de stockage et de requête idéale pour OWL, ni même pour RDFS. Les connaissances dans OWL sont représentées sous la forme d'ensembles de triplets <sujet, prédicat, objet>, l'interrogation de ce documents triples s'effectue via des langages de requête spécifiques le langage standard actuel pour l'interrogation de RDF-S et également interrogation de OWL est SPARQL.

"SPARQL Protocol and RDF Query Language (SPARQL) est utilisé pour exprimer des requêtes sur diverses sources de données, que les données soient stockées de manière native en tant que RDF ou visualisées en tant que RDF-S. SPARQL contient des fonctionnalités permettant d'interroger les modèles de graphique requis et facultatifs, ainsi que leurs conjonctions et leurs disjonctions. SPARQL est un protocole qui permette soumettre des demandes via HTTP GET, HTTP POST ou SOAP. Les résultats des requêtes SPARQL peuvent être des ensembles de résultats ou des graphes RDF (format XML), et maintenant JSON. " [33]

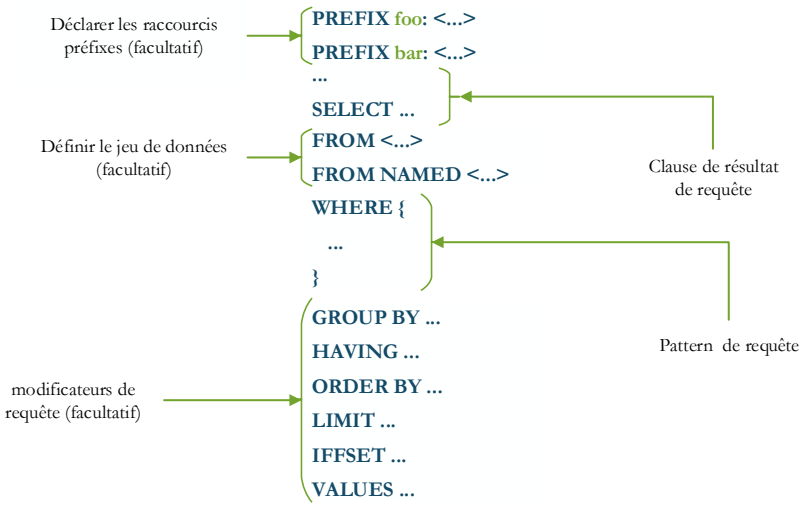


Figure II.12: structure d'une requête SPARQL

La requête commence par les instructions PREFIX qui définissent les abréviations des espaces de noms.

```
REFIX foaf: <http://xmlns.com/foaf/0.1
```

La requête commençant dans sa première ligne par **SELECT**, qui contient également une variable commençant par le caractère de point d'interrogation '?'. Choisissez le mot de votre choix pour le reste du nom de la variable, à condition que vous l'utilisiez de manière cohérente. Le reste de la requête, commençant par **WHERE**, contient une liste de modèles triples RDF. Ce sont comme des triplets RDF sauf qu'ils incluent des variables. Ils sont exprimés dans RDF Turtle.

La réponse à une requête est calculée par un processus appelé « graph matching », illustré schématiquement à la figure sous dessous, où requête et jeu de

données sont présentés sous forme de graphes RDF spécifiés dans RDF Turtle (pour simplifier, seule une partie du jeu de données ci-dessus est incluse).

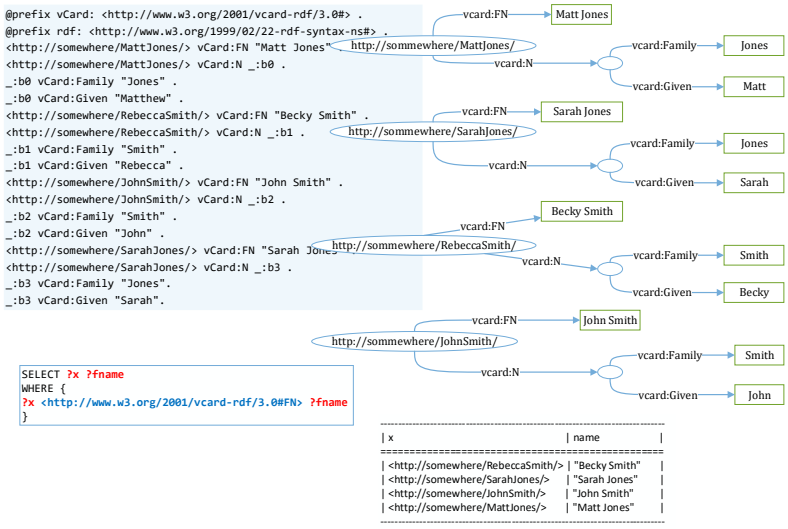


Figure II.13 : une simple requête et Répondre à cette requête par "graph matching"

Types de requêtes

- SELECT : renvoie une table de résultats
- ASK : renvoie (booléen) true, si le motif peut être mis en correspondance
- CONSTRUCT : crée des triples à l'aide de modèles
- DESCRIBE : renvoie les descriptions des ressources

I.7 Le langage SWRL

"SWRL (Semantic Web Rule Language) est un langage de règles Web sémantique combinant les sous-langages OWL DL du langage ontologie Web OWL2 avec les sous-langages Unary / Binary Datalog RuleML du langage de marquage. Les utilisateurs peuvent écrire des règles qui peuvent être exprimées en termes de concepts OWL afin de fournir des capacités de raisonnement déductif plus puissantes que celles d'OWL seul qui permet de créer de nouvelles connaissances à partir des bases de connaissances OWL existantes. Les règles SWRL stockées dans une base de connaissances OWL assurent l'interopérabilité entre la multitude de systèmes, ce qui constitue l'un des principaux objectifs du Web sémantique. " [27]

Les règles SWRL sont écrites comme des paires antécédents-conséquents. Dans la terminologie SWRL, l'antécédent est appelé le corps de la règle (body) et le conséquent est appelé la tête (head). La tête et le corps sont constitués d'une conjonction d'un ou de plusieurs atomes.

Atom ^Atom ^ Atom ... → Atom ^ Atom ...

antécédents conséquents

Un atome est une expression de la forme:

p(arg1, arg2,..., argn)

p : prédicats binaires ou unaire(classes, propriétés ou types de données OWL)

arg1, arg2,..., argn : arguments de l'expression (individus OWL ou valeurs de données, ou variables les référant)

Figure II.14 : La structure de rôle SWRL

Les règles SWRL motivent les individus OWL, principalement en termes de classes et de propriétés OWL. Par exemple, une règle SWRL exprimant qu'une personne ayant un frère ou une soeur a un frère requiert de saisir les concepts de «personne», «homme», « Enfant mêmes parents» et «est frère de» dans OWL. Intuitivement, le concept de personne et d'homme peut être capturé à l'aide d'une classe OWL appelée Personne avec une sous-classe Homme, les relations entre Enfant mêmes parents et est frère de peuvent être exprimées à l'aide des propriétés OWL Enfant_mêmes_parents et est_frère_de, qui sont attachées à Person. La règle dans SWRL serait alors :

$$Personne(?x1) \wedge Enfant_mêmes_parents(?x1, ?x2) \wedge Homme(?x2) \rightarrow est_frère_de(?x1, ?x2)$$

Les règles SWRL peuvent également faire explicitement référence aux individus OWL. Par exemple, l'exemple suivant est une variante de la règle ci-dessus, déduisant qu'un individu particulier Djamel a un frère :

$$Personne(Djamel) \wedge Enfant_mêmes_parents(?x1, ?x2) \wedge Homme(?x2) \rightarrow est_frère_de(Djamel, ?x2)$$

SWRL prend également en charge les littéraux. Par exemple, en supposant qu'un individu possède une propriété Age, il est possible de demander si Fred a un frère de 40 ans :

$$Personne(Djamel) \wedge Enfant_mêmes_parents(?x1, ?x2) \wedge Homme(?x2) \wedge age(?x2,40) \rightarrow a_frère_de_40_an(Djamel, ?x2)$$

SWRL prend également en charge les concepts communs identiques et différents. Par exemple, l'atome SWRL sameAs peut déterminer si deux individus OWL, Djamel et Djamel-eddine, sont le même individu :

$$sameAs(Djamel, Djamel_eddine)$$

SWRL a également un atome pour déterminer si un individu, une propriété ou une variable est d'un type particulier. Ces atomes forme Une plage de données consiste en un nom de type de données ou un ensemble de littéraux et un argument unique représentant une valeur de données.

$$xsd:int(?x)$$

$$[3,4,5](?x)$$

SWRL prend également en charge une gamme des éléments intégrés définis par l'utilisateur, ces éléments intégrés sont des prédicats qui prend un ou plusieurs arguments et ils sont évalués à vrai si les arguments satisfont le prédicat.

Les éléments intégrées SWRL de base sont précédées du qualificateur d'espace de nommage swrlb.

$$Personne(?p) \wedge age\ (?p, ?age) \wedge swrllb{:}puls_age_que(?age, 17) \rightarrow Adulte(?p)$$

Il existe plus de types et de structures d'atomes, on a présenté les plus élémentaires. Enfin, comme mentionné ci-dessus, SWRL est basé sur OWL DL et partage sa sémantique formelle. Il est plus expressif que OWL DL. Cependant, il faut s'efforcer de rester dans OWL si possible et d'utiliser SWRL uniquement lorsque son pouvoir expressif supplémentaire est requis.

I.8 Outils du web sémantique

Il existe de nombreux outils logiciels liés au Web sémantique. Certains éditeurs importants du Web sémantique ont conçu divers types de logiciels pour la création et la manipulation d'ontologies. De nombreux éditeurs d'ontologies sont disponibles sur Internet. Certains d'entre eux, comme Apollo, OntoStudio, Protégé, Swoop et TopBraid) sont utilisés par un très grand nombre de personnes. Le tableau suivant montre sartan caractéristique de ces outils

Tableau II.3: Outils du Web sémantique les plus utilisés

Nom	Editeur	Import format	Export format	License
Apollo	KMI (Open University)	Apollo Meta language	OCML, CLOS	Open-source
OntoStudio	IBM Eclipse framework	XML(S), OWL, RDF(S), UML Diagram, database schemas (Oracle,MSSQL, DB2,MySQL) , Outlook, file system Metadata, Remote OntoBroker	XML(S), OWL, RDF(S),UML,O XML.	Payante
Protégé	SMI (Université de Stanford)	XML(S), RDF(S), OWL, (RDF,UML, XML)backend , Excel, BioPortal , DataMaster	XML(S), RDF(S), OWL, Clips, SWRL-IQ, MetaAnalysis, OWLDoc, Queries, et (RDF, UML, XML) backend	Open-source

Swoop	MND (Université de Maryland)	OWL, XML, RDF et format texte	RDF (S), OIL, DAML	Open-source
TopBraid	Top Quadrant	RDFa, WOL, RDF(s) ,XHTML, Microdata et RDFa Data sources, SPIN, News Feed, RDF Files into a new TDB, Email et Excel	Merge / Convert RDF Graphs, RDF(S), WOL	Payante

I.9 Conclusion

Le web sémantique est une représentation graphique déclarative qui peut être utilisée pour représenter la connaissance et prendre en charge des systèmes automatisés permettant de raisonner sur la connaissance dans des modèles de nœuds et d'arcs interconnectés. Cette représentation repose sur une architecture en couche pour arriver finalement à construire une ontologie qui est conçu pour l'acquisition automatique des connaissances dans l'objectif d'ajouter des novelles concepts, actualiser l'existants et l'interroger. Cette possibilité de raisonner sur les ontologies et les interroger permet de localiser les éléments reliés à un individu donné et de faire des analyses sur les connaissances représentées.

Chapitre III

Analyse de l'impact de changement

III.1 Introduction

Proposer et gérer des modifications a toujours été un processus vital dans le développement de n'importe quels projets. Traditionnellement, il est effectué plus ou moins de manière improvisée. Tout d'abord, un changement est demandé, par exemple par le partie prenante (stakeholder), où le réalisateur de projet utilise son expérience et sa connaissance pour identifier les parties affectées par ces changements, la manière dont ils sont affectés et la manière dont ils doivent être mis à jour.

Dans les petits projets, cette approche peut être appliqué avec prudence. Cependant, la taille et la complexité des systèmes rendent la gestion des changement coûteuse et longue Pour réduire le coût des changements, il est important d'appliquer la gestion des changements dans le cycle de développement de n'importe quel système. Lorsque des changements des exigences sont proposées, l'impact de ces changements sur d'autres exigences et sur tous les éléments de conception doit être analysé afin de déterminer les parties du système à modifier. Cette approche est appelée analyse d'impact de changement.

III.2 Impact de changement

L'importance d'analyser l'impact des changements réside dans leur capacité à évaluer correctement l'ensemble de changements demandés et pour estimer les risques et les coûts associés à la mise en œuvre des changements. En plus d'être une bonne pratique générale, l'analyse d'impact de changement est souvent obligatoire pour les systèmes critiques pour respecte les dispositions de conformité des normes.[39]

Afin d'étudier l'analyse d'impact du changement, nous devons d'abord déterminer certains concepts.

Concept élémentaire – tout élément présent dans le formalisme SysML les diagrammes, les blocks, les attributs, les relations, les ainsi que leurs propriétés telles que le nom, le type, etc.

Relation élémentaire – une relation définie sur l'ensemble de concepts élémentaires induisant une forme de dépendance entre deux concepts élémentaires. Le sens de la relation est donné par le sens de cette dépendance.

Evénement impactant – Evénement ayant pour cible directe un élément impactable, et dont la conséquence est de marquer ce dernier de l'étiquette «impacté». Les événements impactant 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.

Impact – désigne la conséquence de la survenue d'un événement impactant. Ainsi, l'impact d'un événement impactant correspond à l'ensemble des éléments impactables marqués de l'étiquette « impacté ».

Degré d'impact – représente sur une échelle de 0 à 100. Permet d'établir une mesure estimative de l'effort nécessaire pour mettre à jour l'élément pour palier à cet impact. On peut également prendre en compte le type d'impact et le type de l'élément impacté : supprimer une relation représenter un impact plus coûteux que la modification même relation.

Conviction d'impact– représente sur une échelle de 0 à 100 la précision avec laquelle nous pouvons affirmer la présence d'un impact. La valeur 0 correspond au cas où nous ne pouvons pas déterminer s'il existe un impact. La valeur 100 correspond quant à elle au cas où un impact a été identifié à l'aide des données fournies. Les valeurs intermédiaires peuvent nuancer cette conviction en fonction de notre connaissance du domaine : 75 plus de conviction que dans le cas de 50. Par analogie avec la logique flue nous essayons ainsi de mesurer avec des heuristiques la certitude qu'un impact ait lieu ou non.

Elément impacté – est un élément « impactable » qui subit un impact avec un degré et une conviction d'impact supérieurs à zéro.

Etude d'impact – est le calcul du degré d'impact et de la conviction d'impact pour tous les éléments impactbales du modèle et suite à un événement impactant.

Les modalités de propagation des événements impactants au sein d'un modèle dépendent étroitement de la nature des relations définies entre ses éléments impactables.

Ainsi, si le block B est une spécialisation du block A, la modification du nom de B n'engendre aucun impact sur A : la propagation des événements s'effectuant dépendamment de l'orientation des relations.

III.3 Le processus d'analyse l'impact du changement

L'importance d'analyser l'impact des changements réside dans leur capacité à évaluer correctement l'ensemble de changements demandés et pour estimer les risques et les coûts associés à la mise en œuvre des changements. En plus d'être une bonne pratique générale, l'analyse d'impact de changement est souvent obligatoire pour les systèmes critiques pour respecte les dispositions de conformité des normes.

Décrit un processus de géré impact du changement. Un aperçu simple de ce processus, tel que décrit ici, est présenté dans la Fig III.1. Cette mission est dans le genre processus général sous forme de trois éléments : "Rechercher les conditions requises", "Rechercher les conditions dépendantes" et "Proposer des modifications".

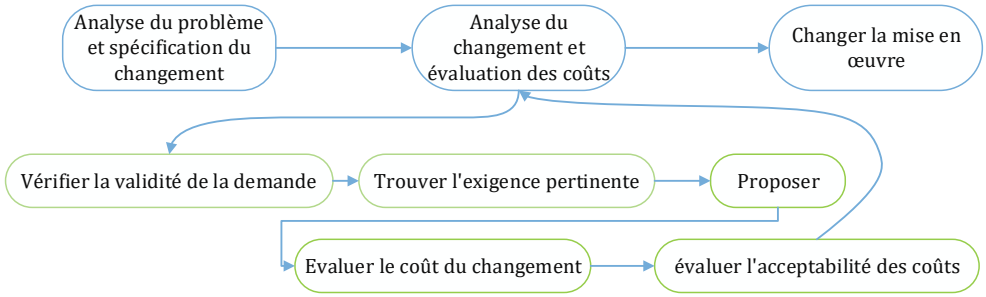


Figure III.1: processus de gestion d'impact de changement [9]

III.4 L'analyse de l'impact des changements dans le contexte SysML

Le processus d'automatisé d'analyse d'impact des modifications dans le contexte des modélisation SysML. Il vise à maintenir synchronisés plusieurs diagrammes de même système en enregistrant les modifications apportées à un diagramme et ses diagrammes associées. Le processus comprend quatre sous-processus :

- Vérifiez la cohérence des diagrammes modifiés. En d'autres termes, vérifiez si les diagrammes d'un système est cohérent avec lui-même
- Effectuer une analyse d'impact, Basé sur les règles d'analyse d'impact.
- Donner la priorité aux résultats de l'analyse d'impact.

Plusieurs approches ont été développées. Cependant, la plupart des approches sont basées sur l'analyse de l'impact des modifications vers et depuis la mise en place de système. Ces approches ont certes leurs avantages, mais dans les systèmes plus complexes, il est également nécessaire de gérer correctement les changements dans les niveaux d'abstraction supérieurs, tels que la conception et les exigences.

L'une des approches qui se concentre principalement sur les exigences est *Semantics of trace relations in requirements models for consistency checking and inferencing*, Cette recherche définit son propre métamodèle d'exigences et ses relations avec une base mathématique bien définie. Ils appliquent une distinction entre les modifications basées sur la notion de domaine d'application. Le domaine est ce que les intervenants ont un intérêt et ce que le modèle doit représenter. Les types de changements sont les suivants :

- Un changement qui est rendu nécessaire par un changement de domaine. Celles-ci affectent les propriétés fonctionnelles et non fonctionnelles du système requis.
- Un changement qui n'est pas lié à un changement de domaine. Ceux-ci n'affectent pas les propriétés fonctionnelles ou non fonctionnelles du système requis.

Ce sont généralement des modifications de re-affacturage.

Lorsqu'une modification est apportée de l'un ou l'autre type, les relations d'exigences sont utilisées comme traces pour déterminer si la modification doit être propagée à d'autres éléments de modèle. Cette propagation est effectuée à l'aide de

règles de propagation, basées sur la définition mathématique des exigences, des relations entre exigences et des modifications.

Les outils d'analyse automatique l'impact de changement actuels qui soutiennent cette approche. IBM Rational "RequisitePro et DOORS" [41]. Bien que ces outils représentent explicitement les relations entre les exigences, les informations fournies sont souvent trop générales et les résultats de l'analyse peuvent ne pas être satisfaisants. Par exemple, dans RequisitePro la modification est limite dans les exigences seulement. Lorsqu'une exigence est modifiée, toutes les exigences accessibles depuis l'exigence modifiée sont potentiellement affectées. Sans information sémantique détermine précisément le degré d'impact de ce changement. Ce type d'analyse produit souvent un grand nombre d'éléments faussement positifs impactés.

Cette situation comme une explosion d'impacts sans sémantique. Il déclare que l'analyse de l'impact du changement doit utiliser des informations sémantiques supplémentaires pour augmenter la précision en éliminant les faux positifs.

III.5 L'importance de la sémantique pour l'analyse d'impact du changement [15]

Le principal défi dans l'analyse de l'impact du changement est de trouver exactement les éléments qui nécessitent un changement et de minimiser le nombre de faux positifs. La fig III.2 présente un exemple des exigences et de l'architecture avec certains liens de trace modélisés. Il existe des relations entre les exigences entre les exigences et des relations entre les composants architecturaux et les exigences qu'elles implémentent. Supposons une demande de modification dans texte de l'exigence R3 en ajoutant la nouvelle contrainte par exemple. Après la mise à jour de R3. En suivant des relations directes et indirectes sans signification, on constate que toutes les exigences (R1, R2, R3, R4, R5 et R6) et les éléments architecturaux (B1, B2 et B3) sont suspects de l'impact.

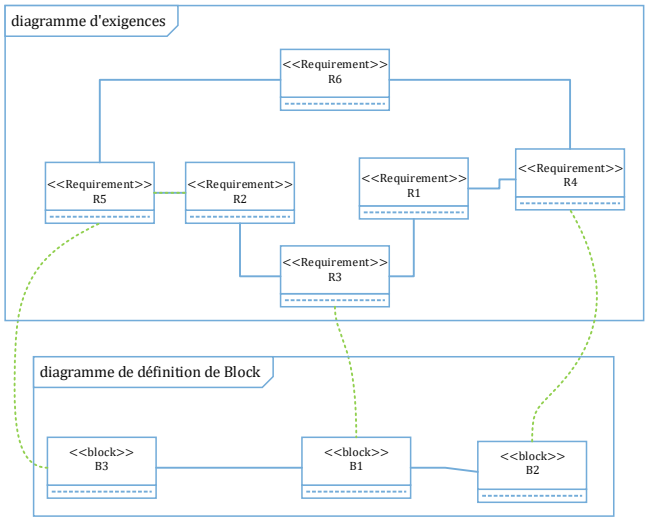


Figure III.2 : exemple d'une partie d'exigences et architecture d'un système

Grâce aux relations entre les éléments impacts, certains suspects sont identifiés comme des faux positifs. La sémantique sert à ajoute une des Règles dans un première temps pour mesure le degré d'impact. Par la suit réduire L'ensemble d'éléments impactés. Par conséquent, établir un raisonnement pour que bien détermine les candidats d'impact des changements.

III.6 Similarité entre textes

Plusieurs techniques sont mises en place pour évaluer la similarité entre documents textuels. Calculer les similarités varie bien évidemment selon les disciplines, mais elles s'intègrent cependant le plus souvent dans une même approche générale en deux temps :

1. Les documents textuels sont d'abord associés à des représentations spécifiques qui vont servir de base au calcul des similarités.

2. Un modèle mathématique est choisi pour mesurer les similarités.[38]

III.6.1 Similarité syntaxique

Une mesure de comparaison textuels, consiste à comparer des chaines de caractères. C'est une métrique qui mesure la similarité ou la dissimilarité entre deux chaînes de caractères. Une telle mesure sur les chaines de caractères fournit une valeur obtenue algorithmiquement.

Parmi de telles mesures de similarité, citons par exemple, la distance de Levenshtein (ou distance d'édition), le coefficient de Dice, l'indice de Jaccard, la distance euclidienne, le cosinus.[26]

L'avantage de ces techniques est qu'elles reposent sur l'approche syntaxique et ne laissent aucune place aux exceptions. Donc ils sont donc faciles à automatiser.

Par contre ils ne prennent pas en compte la sémantique. Par exemple, il est difficile de trouver une forte similitude entre "je possède un chien" et "j'ai un animal". Par conséquent, la prise en compte de la sémantique semble importante. Par conséquent, il faudrait trouver un moyen d'incorporer variation du texte. De même, les rôles sémantiques sont ignorés. Par exemple, Problèmes négatifs (par exemple, "je suis malade" et "je ne suis pas malade") et les antinomies semblent encore difficiles à vaincre.

III.6.2 Similarité sémantique

La similarité sémantique est un concept selon lequel un ensemble de documents ou de termes se voient attribuer une métrique basée sur la ressemblance de leur signification / contenu sémantique. Concrètement, cela peut être réalisé en définissant une similitude topologique, par exemple, en utilisant des ontologies pour définir une distance entre les mots, ou en définissant une similitude statistique, par exemple en utilisant un modèle d'espace vectoriel pour corréler les termes et les contextes à partir d'un corpus de texte approprié (co-occurrence).

Parmi de telles mesures de similarité deux approches. Les approches vectorielles où ils consistent à déterminer la sémantique d'un mot en consultant les autres termes utilisés à ses côtés dans des phrases, comme mesure en peut cites Bi-clustering. Les approches topologiques, ces approches de similarité de mots basées sur la connaissance s'appuient sur un réseau sémantique de mots, tel que WordNet, Etant donnés deux mots, leur similarité peut être estimée à partir de leurs positions relatives dans la hiérarchie de la base de connaissances.

Comparativement aux approches syntaxiques, certains inconvénients ont été palliés comme les problèmes liés à la négation et l'antinomie, aux rôles sémantiques inverses. De plus, il est à noter que les approches sémantiques basées sur la connaissance posent des problèmes de stockage et de complexité et sont souvent spécifiques à un domaine donné.

Ils existent plusieurs techniques pour chaque un de ces deux types de similarité textuelle. La décision finale de la technique à développer doit être prise selon le conteste de problème donne.

III.7 Conclusion

Dans ce chapitre, nous avons présenté l'analyse de l'impact de changement et détailler les terminologies employées dans l'étude d'impact au sein d'un modèle d'un système. Par la suit, nous avons expliqué les différentes approches de mesure de la similarité textuelle et sémantique. A cause de vitalité de faire des modifications au cours des progressions d'un système, plusieurs approches sont mises en place pour capturer et classer l'impact de changement. L'utilisation de la sémantique formelle des types de relation conditionnelle donne une meilleure compréhension pour bien analyser l'impact de changement.

L'exploitation de telle formalisation dite sémantique donne la possibilité de mesurer la similarité textuelle entre les nouvelles exigences et celles que nous voulons changer afin de bien déterminer le degré d'impact ainsi que les éléments impactés.

Chapitre IV

Méthodologie et Conception

IV.1 Introduction

Afin d'améliorer le niveau sémantique du modélisation SysML, les ontologies présentent une solution prometteuse qui nous aidera non seulement de compléter la cohérence entre les différents éléments du système, mais également à trouver des techniques analytiques permettant d'obtenir des informations pertinentes sur un système pour un besoin spécifique, ainsi qu'une méthode de raisonnement pour mesurer le degré d'impact en cas de changement.

Dans ce qui suit, nous présentons la spécification de la modélisation SysML, et nous décrivons l'approche en général. Par la suite, nous expliquons et nous détaillons les différentes phases de notre approche.

IV.2 La spécification de modélisation en SysML

Le langage SysML devient une solution prometteuse pour la conception de systèmes comprenant des systèmes complexes et critiques. Il propose des concepts utilisables dans plusieurs domaines à différents niveaux d'abstraction. Grâce à la diversité des diagrammes, les vues système se varient, ce qui nous permet de mettre en œuvre une conception globale du système.

De point de vu exigences, elles sont couramment utilisées pour formaliser les prérequis du système, se traduisant par des besoins ou des règles qui doivent être satisfaites ou des critères de performance à atteindre.

SysML fournit des éléments pour représenter des exigences textuelles et les relier à d'autres éléments de modélisation. 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>>), Une exigence peut contenir des sous-exigences dans une hiérarchie d'exigences (<<decompose>>), SysML présente le concept d'une exigence esclave. Une exigence esclave est une exigence dont la propriété de description textuelle est une copie (stéréotype <<copy>>) en lecture seule de la même propriété d'une exigence maître. La relation <<satisfy>> décrit comment une conception ou un modèle de mise en œuvre satisfait une ou plusieurs exigences. Par exemple, la conception du moteur satisfait l'exigence de puissance du moteur. La relation <<verify>> définit comment un scénario de test ou d'autres éléments de modèle vérifient une exigence. De même, la relation d'exigence

<<refine>> peut être utilisée pour décrire comment un élément de modèle peut être utilisé pour affiner une exigence. Par exemple, un cas d'utilisation ou un diagramme d'activité peuvent être utilisés pour raffiner une exigence fonctionnelle.[35]

De point de vu structure, le « bloc » est une structure modulaire dans SysML utilisée pour définir les types d'entités physiques (système, composant de système, systèmes externes ou éléments circulant dans le système), ainsi que des entités conceptuelles ou des abstractions logiques.

"Le diagramme de définition de bloc (bdd) est utilisé pour définir les caractéristiques de chaque bloc en termes de caractéristiques structurelles et comportementales, ainsi que de relations entre les blocs, telles que des relations (<<association>>) permettre de définir le flux d'informations entre composants, la spécification entière peut être décomposée en sous blocks, qui peuvent être elles-mêmes décomposées et ainsi de suite pour définir la hiérarchie de block par la relation (<<composition>>), utiliser la relation (<<aggregation>>) si la composition ne nécessite indispensable entre les blocks. [34]

De point de vu comportements, il est composé d'un ensemble d'actions exécutées avec un ordre de vocation spécifique. Ce dernier est imposé par les flux de contrôle, accentuant éventuellement les dépendances d'entrée et de sortie à l'aide de flux de données. Les actions peuvent être coordonnées séquentiellement ou simultanément. De plus, le diagramme peut impliquer une synchronisation et / ou une ramification. Ces fonctionnalités sont activées à l'aide des nœuds de contrôle prédéfinis prenant en charge diverses formes de routage de contrôle. La simultanéité et la synchronisation sont modélisées à l'aide de forks et de jointures, alors que les branches sont modélisées à l'aide de nœuds de décision et de fusion [19].

Le diagramme des exigences est le support principal de SysML pour la traçabilité entre les exigences ainsi que pour la traçabilité des exigences aux structures et comportements dans le modèle de système. Au fur et à mesure que on ajouter de nouveaux éléments au modèle, on créer des relations entre ces éléments et les exigences qui en ont nécessité la création. Établir la traçabilité des exigences de cette manière est une activité continue tout au long de la conception et du développement. [25]

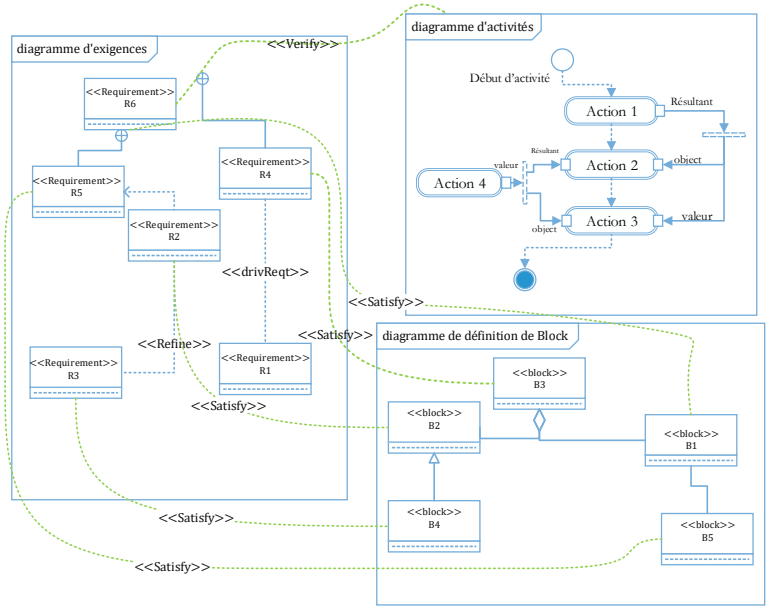


Figure IV.1 : Les informations de traçabilité requises par les 3 diagrammes d'exigences, définition de Block et Activités

Le processus de développement de n'importe quel système ne dépend pas seulement sur la partie conceptionnelle mais aussi sur la structure organisationnelle dans la mesure de la communication entre l'équipe de conception et les différentes équipes de développement système. En raison de l'ambiguïté causée par la modélisation semi-formelle car telle représentation graphique contient trop de détails et souvent une telle spécification manque de justification et explication. Ou plus les mécanismes de mappage permettent d'explorer bien les éléments des systèmes et les relations entre aux mais en cas de modification, il est difficile de garder la cohérence entre les éléments des différents diagrammes du système, aussi bien qu'il n'y a aucune approche pour l'analyse de l'impact en cas de changement sur les exigences, sur la structure ou sur le comportement du système.

C'est pourquoi une transformation de la spécification SysML en une spécification formelle est nécessaire pour arriver par la suit à analyse d'impact du changement et calculer le degré d'impact d'un changement. Comme tous les éléments de SysML sont de nature textuelle, nous proposons une solution basée sur la similarité textuelle et sémantique entres les éléments de modélisation SysML.

IV.3 L'approche proposée

Notre approche est basée sur trois phases, comme illustre la fig IV.2.

La Phase Formalisation et Transformation : dans cette phase nous définissons une ontologie qui permet de décrire les trois diagrammes SysML : diagramme d'exigences, diagramme de définitions de block et diagramme d'activité. Ceci est fait en se basant sur le méta-modèle du langage SysML. Nous passons par la suite à la transformation des diagrammes SysML vers le langage OWL.

La Phase Analyse : La deuxième phase permet d'enrichir l'ontologie obtenue en phase 1 par des requêtes et des règles pour pouvoir analyser les différents éléments des diagrammes et les relations entre eux. En résultat si on propose de modifier un élément donné, on peut déterminer les éléments impactés par ce changement.

La Phase calcul du degré d'impact : la troisième phase consiste à calculer les degrés d'impact des changements en se basant sur des mesures de similarité sémantique entre un élément de modèle que l'on veut modifier et les éléments impactés.

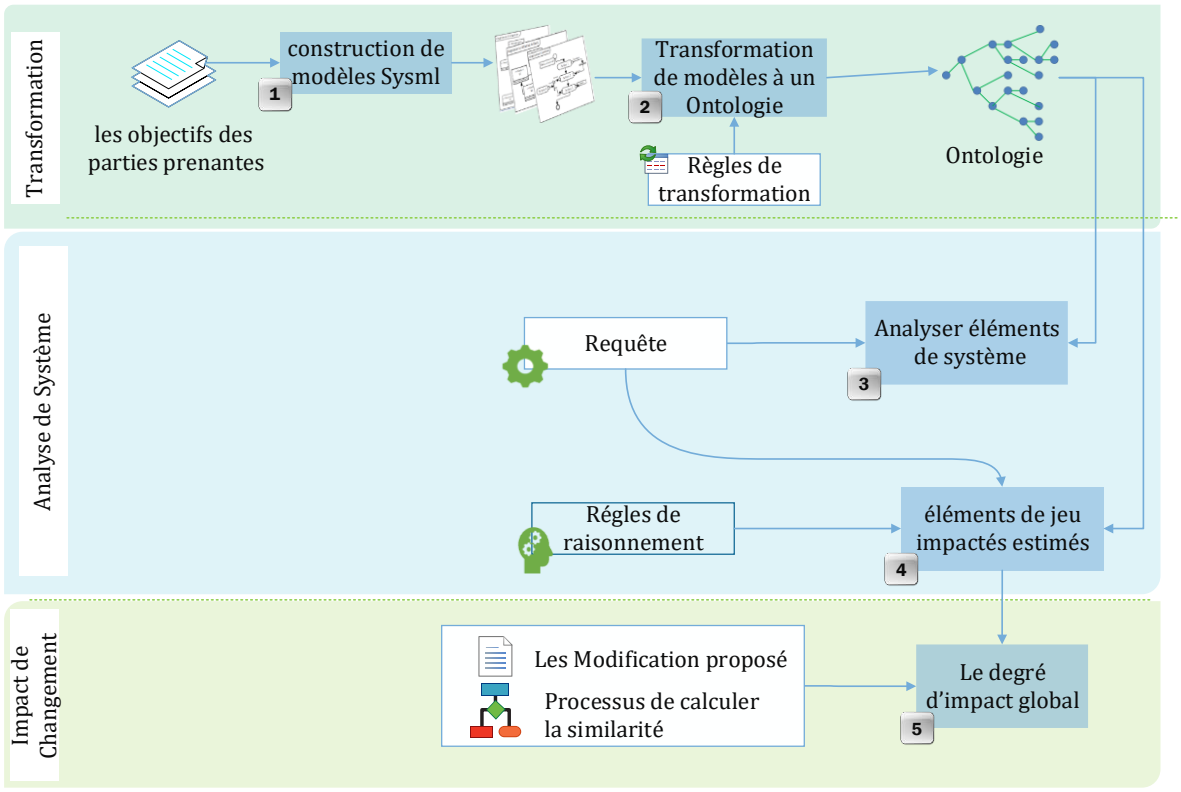


Figure IV.2 : Schéma Synoptique de l'Approche Proposée

IV.4 Conception Détaillée

Nous détaillerons dans ce qui suit la démarche que nous avons proposée.

IV.4.1 Formalisation et Transformation

Cette phase est décomposée en deux étapes la formalisation et la transformation

IV.4.1.1 La Formalisation

Le but est la formalisation des trois diagrammes : Exigences, Bloc et Activité, ainsi que les relations utilisées dans chacun de ces diagrammes. Comme nous l'avons dit précédemment, nous avons choisi une formalisation sémantique en langage OWL. L'expressivité du OWL est basée sur un ensemble de classes (concepts) où une classe peut avoir des sous-classes, comme il peut avoir aussi un ou plusieurs propriété et/ou sous-propriété et un ensemble de prédicats (relations) où chaque relation est associée à deux classes, l'un d'eux "domain" et l'autre "range".

Nous avons cinq (5) concepts principaux : diagramme d'exigences, diagramme de définition de block, diagramme d'activité, diagramme de cas d'utilisation et finalement les parties prenantes (Stakeholder). Les différents concepts inclus dans ces cinq concepts principaux sont définis comme suit :

1. **Diagramme d'exigences :**
   - **Package** : Pour spécifie le type d'un groupe d'exigence ou de l'intégralité de diagramme d'exigence.
   - **Exigence :** Le composant de base du diagramme est caractérisé par
     - **Id** : un identificateur unique ;
     - **Texte** : une description ;
     - **Stéréotype** : une sorte d'étiquetage nommée pour spécifier de quel type d'exigence s'agit-il (fonctionnel ou non fonctionnel)**.**
2. **Diagramme de Définition de Block :**
   - **Package :** Pour spécifie le type d'un groupe d'éléments tels que blocs, acteurs, etc. ou de l'intégralité de diagramme.
   - **Acteur :** Rôle joué par un utilisateur humain ou un autre système
   - **Block :** composant élémentaire de la structure d'un système. Via les stéréotypes en peut distinguer s'il s'agit d'un élément de système (Block), un système ou un sous-système.
   - **Opération :** un bloc peut possède des propriétés comportementales, exemple d'opération, une requête synchrone (l'émetteur est bloqué en attente d'une réponse), une requête asynchrone (l'émetteur n'est pas bloqué en attente d'une réponse).
   - **Référence, Parts, Valeurs et Data type** : sont utilisées pour modéliser les caractéristiques quantitatives des blocs.
3. **Diagramme d'Activité :**
   - **Action :** une action est l'unité fondamentale de spécification comportementale. Elle représente un traitement ou une transformation. Et pour modalise ce qu'un action produit, consommé ou transformé au cours de son exécution en utilise les trois types de pin d'entrée, de sortie et de données
   - **Nœud de control peut être :**
     - **Nœud initial** : recevoir l'ordre au début de l'exécution d'une activité et passer le contrôle au nœud exécutable.
     - **Le nœud Décision** pour représente un choix dynamique entre plusieurs conditions qui doivent être mutuellement exclusives.
     - **Le nœud fork** pour représente un débranchement parallèle. Il duplique le jeton entrant sur chaque flot sortant.
     - **Le nœud join** pour représente une synchronisation entre actions (rendez-vous). Il ne produit son jeton de sortie que lorsqu'un jeton est disponible sur chaque flot entrant
     - **Nœud final :** est le nœud auquel un flux dans une activité s'arrête peut-être soit un nœud final de flux (Flow Final Node) qui termine un flux en consommant les jetons qui lui sont proposés. Soit un nœud final d'activité (Activity Final Nod) qui termine l'exécution d'une activité structurée.

- Nœud d'objet peut être :
  Un nœud de tampon central (Central Buffer Node) qui permettant de gérer les flux provenant de plusieurs sources et cibles. Un nœud de paramètre d'activité (Activity Parameter Node) qui permettant d'accepter des valeurs des paramètres d'entrée ou de fournir des valeurs au paramètres de sortie d'une activité.

4. **Le diagramme de cas d'utilisation :**
   Ce diagramme décrit l'utilisation d'un système par ses acteurs pour atteindre un objectif. Nous le présentons ici sans détailler juste pour montrer la traçabilité de la relation de raffinage, qui liée une exigence et le diagramme de cas d'utilisation
5. **Partie prenante (Stakeholder) :**
   Ce sont les intervenants tout au long de la période de développement du système, soit dans la phase analyse de besoins, conception et réalisation, ou test et validation, ces parties prenantes peuvent être des individus ou des organisations, et c'est à eux de planifier, géré, prendre la décision durant déroulement de projet

La figure ci-dessous résume les classes et leurs sous-classes que nous avons proposés ainsi leurs propriétés.

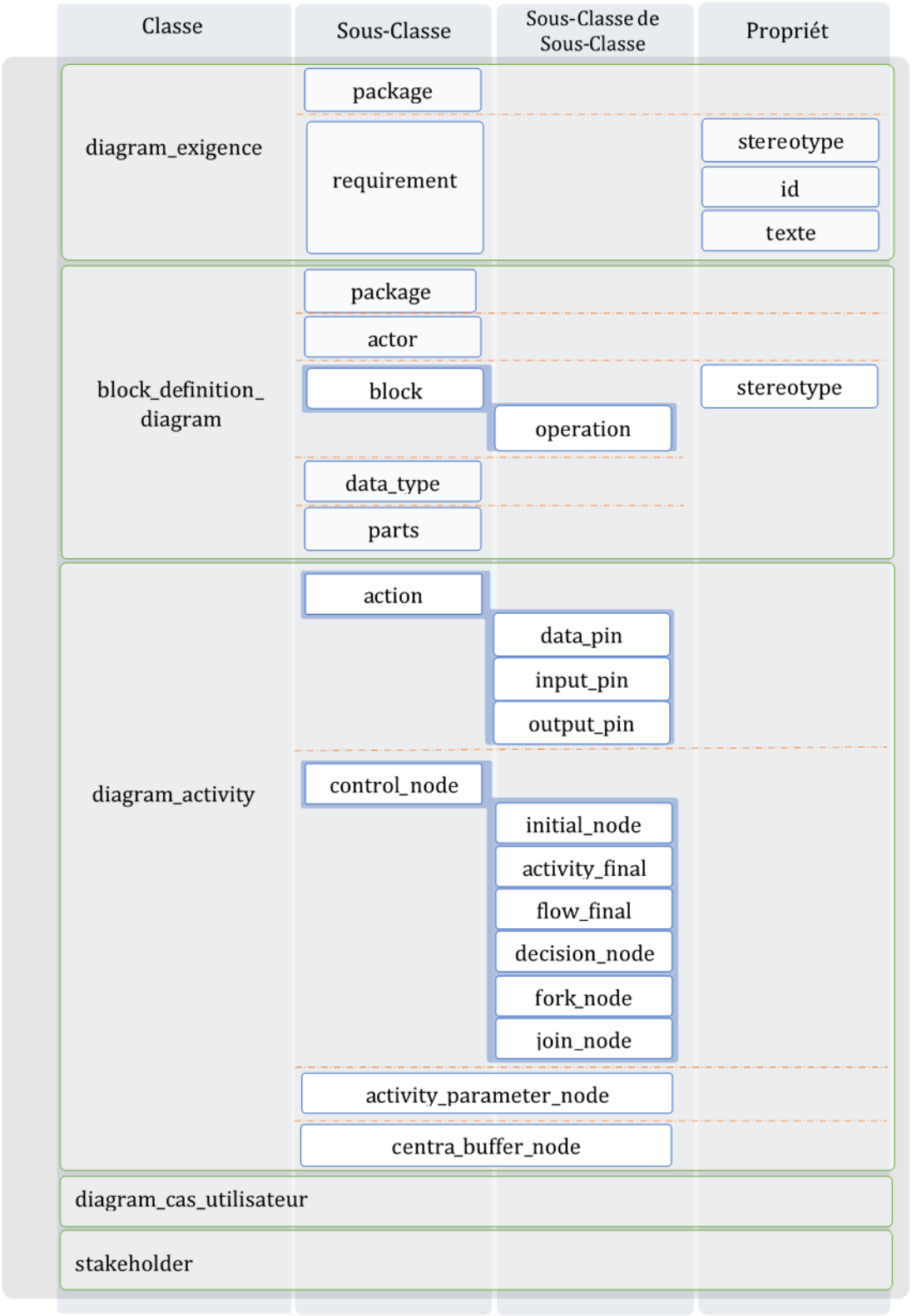


Figure IV.3 : les concepts de l'ontologie proposée pour représenter les diagrammes Sysml

En ce qui concerne les relations entre ces classes (concepts) nous fournissons la formalisation pour la relation Satisfy. Les autres relations sont formalisées de manière similaire.

Nous supposons que le block B doit satisfit l'exigence R. Selon la syntaxe OWL, B est le sujet, R est l'objet et Satisfy est le prédicat. En conséquence le sujet B sera le Domain de la relation et l'objet R sera sa Range.

Sujet	Prédicat	Objet	
B	Satisfy	R	<owl:ObjectProperty rdf:about="#Satisfy"> <rdfs:domain rdf:resource="#block"/> <rdfs:range rdf:resource="#requirement"/> </owl:ObjectProperty>

Figure IV.4 : la formalisation proposée de la relation "Satisfy"

Les relations dans la syntaxe RDF sont Object Properties. Dans notre proposition, nous avons regroupé pour chaque diagramme ces relations dans une relation principal, et pour chacun de ces relations, nous les distinguons leur convenable domain et range. La figure suivante montre en détaille la liste de relations formalisé ainsi que leur domain et range.

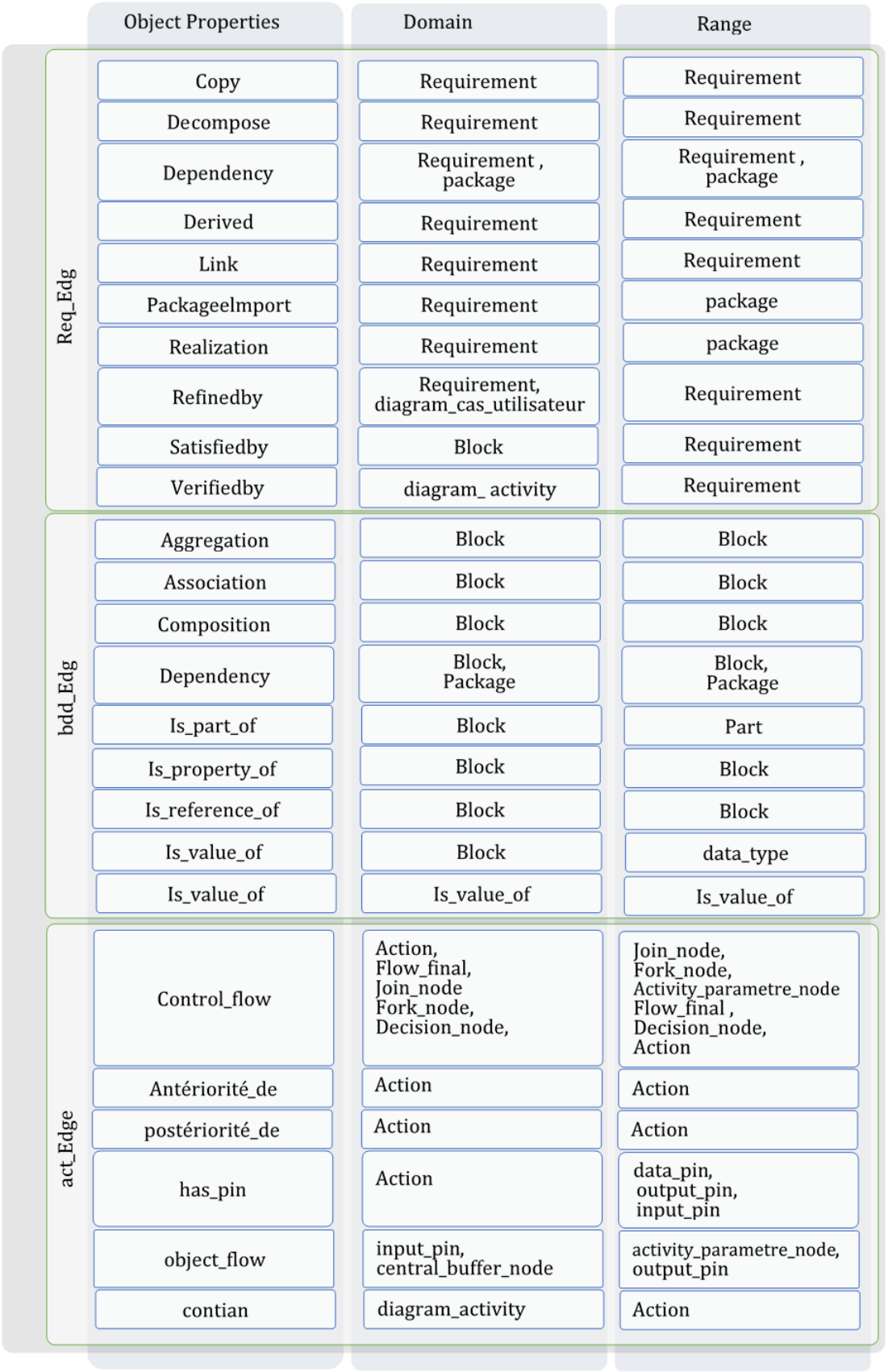


Figure IV.5 : les relations de l'ontologie proposée pour représenter les diagrammes SysML

À la fin de cette étape nous obtenons comme résultat la conception finale des concepts avec lesquelles nous avons créés notre ontologie. Nous entendons par là, premièrement, déterminer le Tbox de l'ontologie qui est une partie fixe qui démontre comment les éléments de différents diagrammes seront représentés. Deuxièmement, avoir une idée de la structure des règles pour créer un moteur de transformation pour générer l'Abox, pour aboutir à une ontologie complète. Ceci est expliqué dans la figure IV.6 suivante.

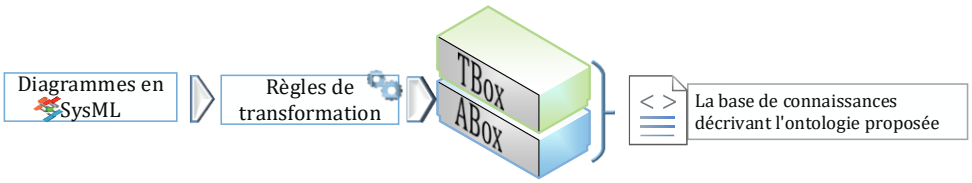


Figure IV.6 : le démarche de création de deux Box de connaissance OWL

IV.4.1.2 La Transformation

Dans cette partie, nous avons proposé un ensemble de règles afin d'extraire les informations à partir du model SysML pour construire un texte structuré qui présente les connaissances en format OWL. Cette approche est faisable à l'aide de transformation M2T (Model to Text). Où la transformation est basée sur la définition des règles qui permette de générer à partir d'un modèle conforme à un méta-modèle, à une structure conforme à notre ontologie déjà proposé.

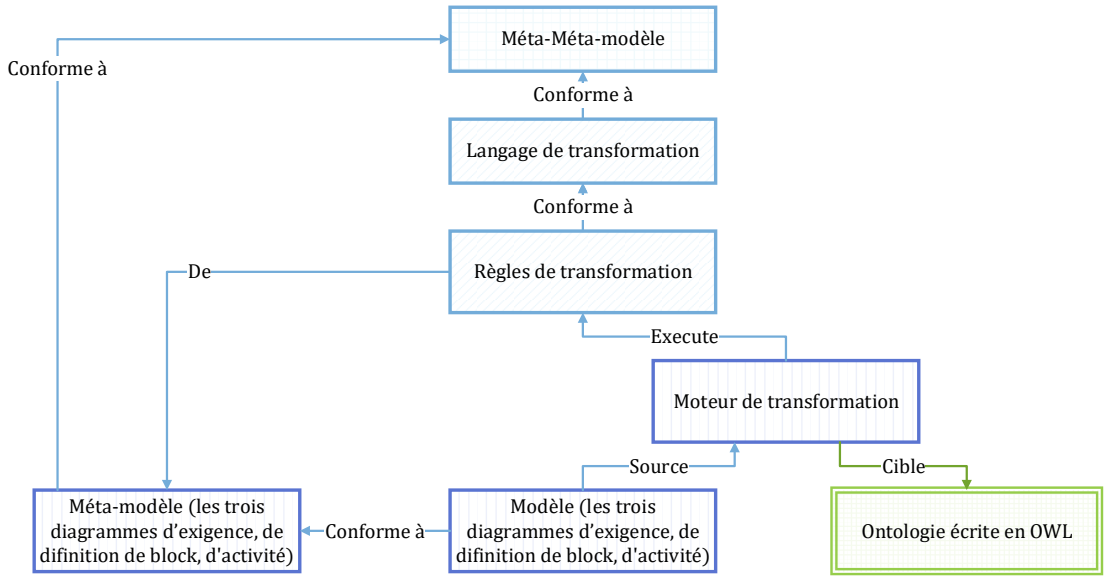


Figure IV.7 : La transformation du modèle SysML vers le langage OWL

La transformation (Model to Text) de chaque concept SysML vers un concept ontologique est faite comme suit :

1. **Diagramme d'exigences :**

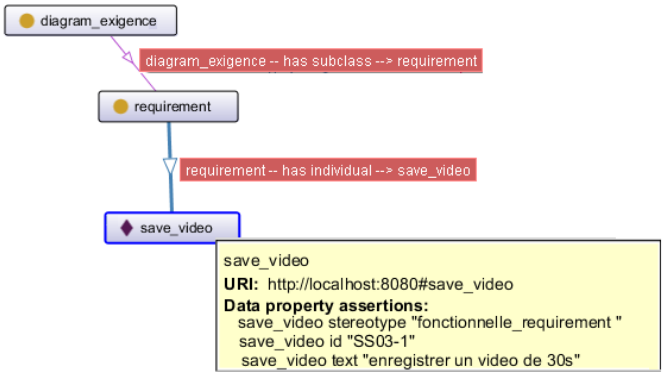
La sous classe "diagram_exigence" est décompose en deux sous classes, package, requirement. Pour transformer une exigence qui a comme stéréotype la valeur "fonctionnelle_requirement" et l'identificateur "SS03-1" et le texte suivant : " enregistrer un vidéo de 30s". La transformation se fait comme suit :

Tableau IV.1 : Transformation d'un Exigence

Forme SysML



Forme ontologique



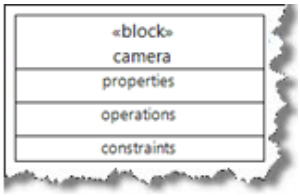
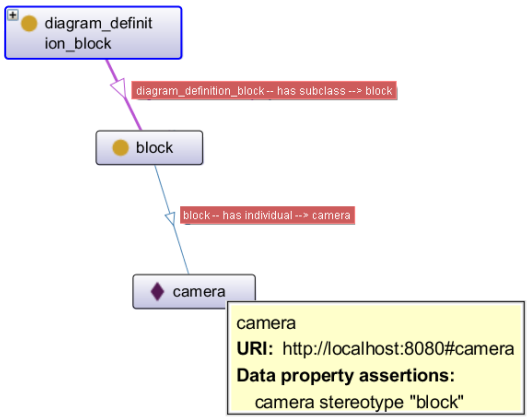
Forme RDF/XML

```
<!-- http://localhost:8080#save_video -->
<owl:NamedIndividual rdf:about="http://localhost:8080#save_video">
 <rdf:type rdf
:resource="http://localhost:8080#requirement"/>
  <id>SS03-1</id>
  <text>enregistrer un video de 30s</text>
  <stereotype>fonctionnelle_requirement</stereotype>
</owl:NamedIndividual>
```

2. Diagramme de Définition de Block

En ce qui concerne la sous-classe “diagram_block_definition” en soi est divisé en cinq sous-classes package, actor, block, datatype, parts. Et en titre d’exemple, la transformation d'un bloc qui prend le stéréotype “block” et « caméra » comme valeur, se fait comme montré dans le tableau Tab IV.2 ci-dessous :

Tableau IV.2 : Transformation d'bloc d'un BDD

Forme SysML	
Forme ontologique	

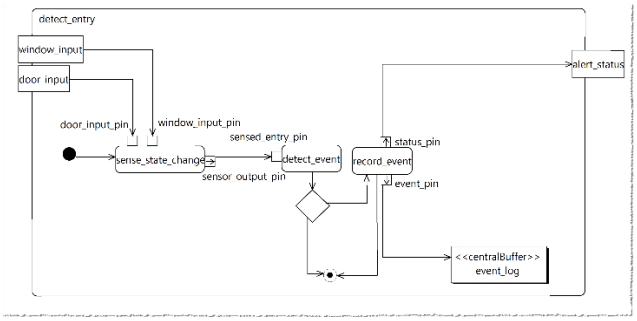
Forme RDF/XML	`<!-- http://localhost:8080#camera -->  <owl:NamedIndividual rdf:about="http://localhost:8080#camera">`
	`<rdf:type rdf:resource="http://localhost:8080#block"/>`
	`<stereotype>block</stereotype>`
	`</owl:NamedIndividual>`

3. Diagramme d'Activité

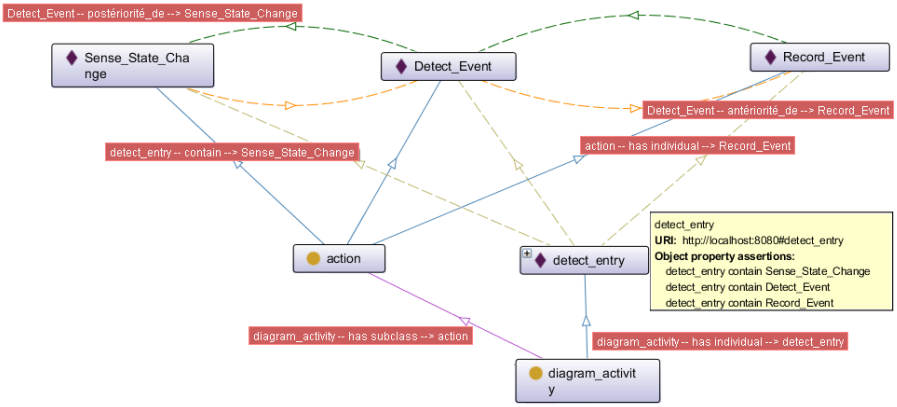
La sous classe "diagram_activity" regroupe les concepts action et leur pins qui contrôle le type de données entrant et sortant, les nœuds de contrôle tel que le nœud initiale, fin d'activité ainsi qui les nœuds d'objets. Dans notre exemple, on va monter une simple description d'un diagramme d'activité nommée "detect_entry" contient trois actions qui sont respectivement "Sense_State_Change", "Detect_Event", "Record_Event".

Tableau IV.3 : Transformation d'un diagramme d'activité

Forme SysML



Forme ontologique



Forme RDF/XML

```
<!-- http://localhost:8080#detect_entry -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#detect_entry">
        <rdf:type
rdf:resource="http://localhost:8080#diagram_activity"/>
        <contain rdf:resource="http://localhost:8080#Detect_Event"/>
        <contain rdf:resource="http://localhost:8080#Record_Event"/>
        <contain
rdf:resource="http://localhost:8080#Sense_State_Change"/>
    </owl:NamedIndividual>
 <!-- http://localhost:8080#Sense_State_Change -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Sense_State_Change">
        <rdf:type rdf:resource="http://localhost:8080#action"/>
        <antériorité_de
rdf:resource="http://localhost:8080#Detect_Event"/>
    </owl:NamedIndividual>
<!-- http://localhost:8080#Detect_Event -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Detect_Event">
        <rdf:type rdf:resource="http://localhost:8080#action"/>
        <antériorité_de
rdf:resource="http://localhost:8080#Record_Event"/>
        <postériorité_de
rdf:resource="http://localhost:8080#Sense_State_Change"/>
```

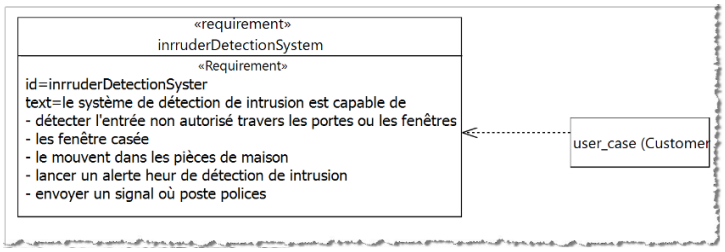
```
        </owl:NamedIndividual>
    <!-- http://localhost:8080#Record_Event -->
        <owl:NamedIndividual rdf:about="http://localhost:8080#Record_Event">
            <rdf:type rdf:resource="http://localhost:8080#action"/>
            <postériorité_de
    rdf:resource="http://localhost:8080#Detect_Event"/>
        </owl:NamedIndividual>
```

4. Le diagramme de cas d'utilisation

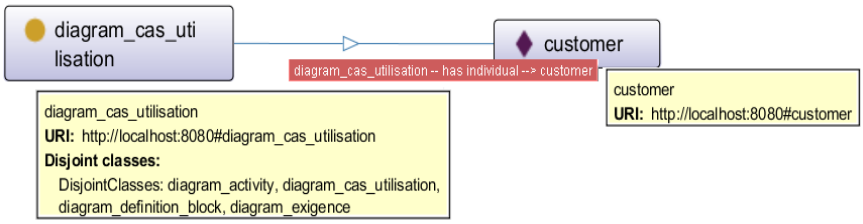
Dans notre approche, nous n'avons pas détaillé le diagramme de cas d'utilisation. Mais, nous l'avons présenté par une simple classe dans le diagramme des exigences pour conserver la traçabilité entre une exigence et le cas d'utilisation qu'elle affine. L'exemple suivant illustre la transformation de ce dernier.

Tableau IV.4 : Transformation d'un cas d'utilisateur

Forme SysML



Forme ontologique



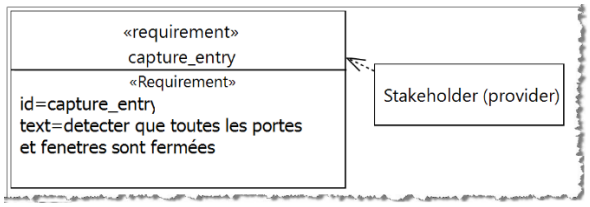
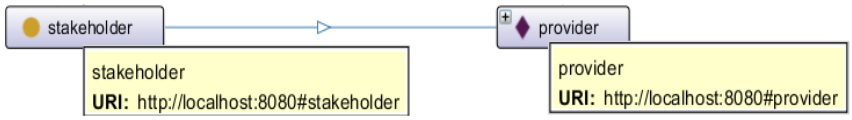
Forme RDF/XML

```
<!-- http://localhost:8080#customer -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#customer">
        <rdf:type
rdf:resource="http://localhost:8080#diagram_cas_utilisation"/>
    </owl:NamedIndividual>
```

5. Partie prenante (Stakeholder)

De même, le Stakeholder est présenté dans SysML par une classe simple dans le diagramme des exigences, il est transformé en un concept ontologique, tel qu'il est présenté dans le tableau Tab IV.5 suivant.

Tableau IV.5 : Transformation d'un stakeholder

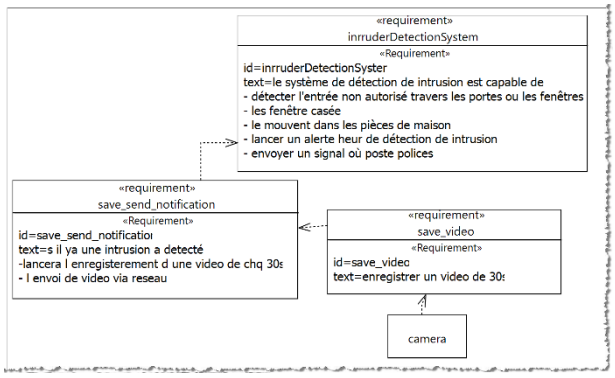
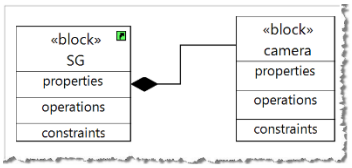
Forme SysML	
Forme ontologiq	

Forme RDF/XML	`<!-- http://localhost:8080#provider -->`
	`<owl:NamedIndividual rdf:about="http://localhost:8080#provider">`
	`<rdf:type rdf:resource="http://localhost:8080#stakeholder"/>`
	`</owl:NamedIndividual>`

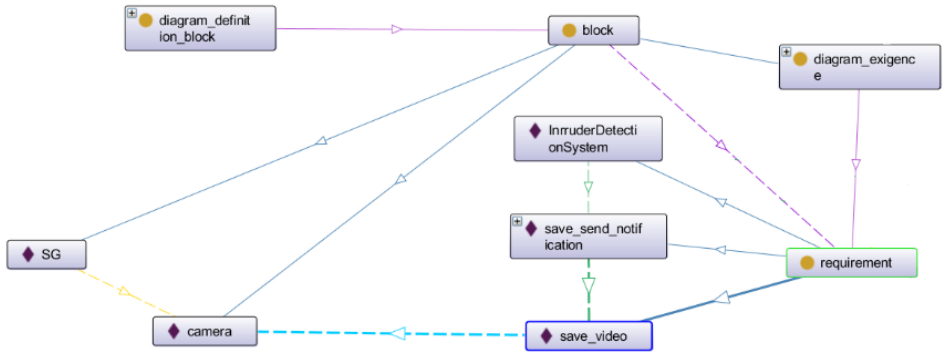
Concernant les relations dans ce qui suit nous avons détaillé deux exemples dans chacun d'eux, nous nous concentrons sur un type de relation.

Premièrement, on suppose que l'exigence "save_video" est une décomposition de l'exigence "save_sand_notification" qu'est dérivée de l'exigence " IntruderDetectionSystem", cette exigence est satisfaite par le bloc "Camera" qui est à sa part une composition d'un système général "SG".

Tableau IV. 6 : Transformation de relation "satisfy" qui liée entre une exigence dans un diagramme d'exigence et un bloc dans un BDD

Forme SysML		
	Diagramme d'exigences	BDD

Forme ontologique



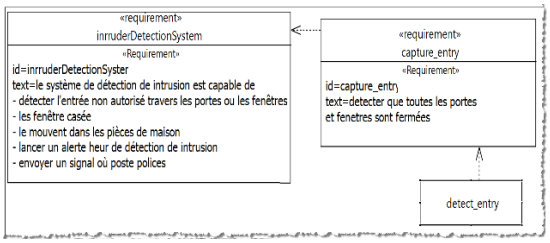
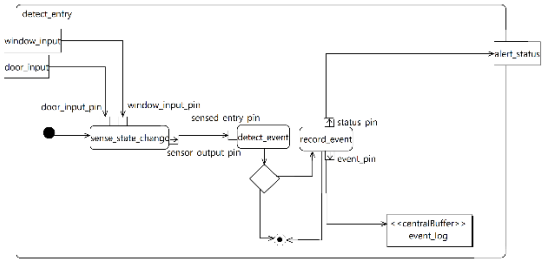
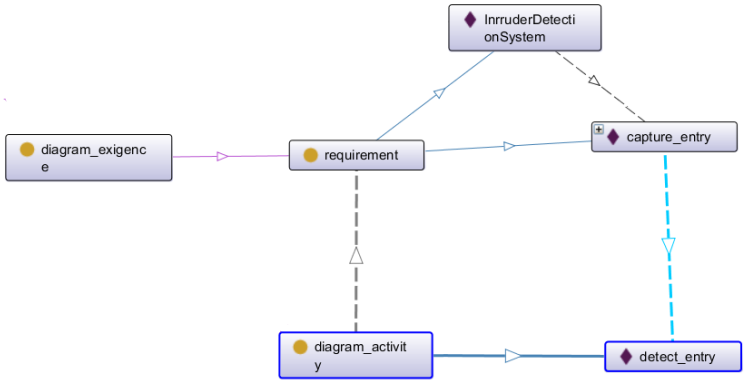
Forme RDF/XML

```
<owl:NamedIndividual
rdf:about="http://localhost:8080#InrruderDetectionSystem">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <id>SG01</id>
        <text>le système de détection de intrusion est capable de
- détecter entrée non autorisé travers les portes ou les fenêtres
- les fenêtre casée
- le mouvent dans les pièces de maison
- lancer un alerte heur de détection de intrusion
- envoyer un signal où poste polices</text>
        <stereotype>requirement</stereotype>
        <derived rdf:resource="http://localhost:8080#alerte_sound"/>
        <derived
rdf:resource="http://localhost:8080#capture_entry"/>
        <derived
rdf:resource="http://localhost:8080#save_send_notification"/>
    </owl:NamedIndividual>
 <owl:NamedIndividual
rdf:about="http://localhost:8080#save_send_notification">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>requirement</stereotype>
        <text>s il ya une intrusion a detecté
-lancera l enregisterement d une video de chq 30s
- l envoi de video via reseau</text>
        <id>SS03</id>
        <decompose rdf:resource="http://localhost:8080#save_video"/>
        <decompose
rdf:resource="http://localhost:8080#send_notification"/>
    </owl:NamedIndividual>
   <!-- http://localhost:8080#save_video -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#save_video">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <id>SS03-1</id>
        <text>enregistrer un video de 30s</text>
```

```
            <stereotype>requirement</stereotype>
            <satisfiedby rdf:resource="http://localhost:8080#camera"/>
        </owl:NamedIndividual>
    <!-- http://localhost:8080#SG -->
        <owl:NamedIndividual rdf:about="http://localhost:8080#SG">
            <rdf:type rdf:resource="http://localhost:8080#block"/>
            <composition rdf:resource="http://localhost:8080#alerte"/>
            <composition  rdf:resource="http://localhost:8080#camera"/>
            <composition
    rdf:resource="http://localhost:8080#mouvement_sensor"/>
        </owl:NamedIndividual>
     <!-- http://localhost:8080#camera -->
        <owl:NamedIndividual rdf:about="http://localhost:8080#camera">
            <rdf:type rdf:resource="http://localhost:8080#block"/>
            <stereotype>block</stereotype>
        </owl:NamedIndividual>
```

Dans le deuxième exemple, on suppose que l'exigence "capture_entry" qui dérivée de principal "IntruderDetectionSystem" est vérifié par l'activité "detect_entry".

Tableau IV.7 : Transformation de relation "verify" qui liée entre une exigence dans un diagramme d'exigence et un Activité

Forme SysML		
	Diagramme d'exigences	Diagramme d'activité
Forme ontologique		

Forme RDF/XML

```
<!-- http://localhost:8080#capture_entry -->
<owl:NamedIndividual rdf:about="http://localhost:8080#capture_entry">
<rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <id>SS01</id>
        <text>detecter que toutes les portes et fenetres sont fermées</text>
        <stereotype>requirement</stereotype>
        <verifiedby rdf:resource="http://localhost:8080#detect_entry"/>
</owl:NamedIndividual>
 <!-- http://localhost:8080#detect_entry -->
<owl:NamedIndividual rdf:about="http://localhost:8080#detect_entry">
        <rdf:type rdf:resource="http://localhost:8080#diagram_activity"/>
        <contain rdf:resource="http://localhost:8080#Detect_Event"/>
        <contain rdf:resource="http://localhost:8080#Record_Event"/>
        <contain rdf:resource="http://localhost:8080#Sense_State_Change"/>
</owl:NamedIndividual>
```

Lorsque nous terminons ce processus de transformation de tous les concepts et relations liés à un système particulier, nous obtenons la partie Abox de l'ontologie. Fusionnée avec la Tbox obtenue dans la phase 1, nous obtenons l'ontologie entière, qui peut être utilisée dans les phases suivantes de notre approche.

IV.4.2 L'analyse sémantique des éléments du système

Dans cette phase, nous prenons le résultat de la phase précédente et via le langage de requête SPARQL, nous implémentons un ensemble de requêtes pour créer un Framework qui va être mis à la disposition de l'utilisateur pour interroger des spécifications SysML.

Le langage SPARQL est un langage basé dans sa partie principale d'interrogation sur le modèle triple de l'ontologie. Ceci est disponible via la représentation RDF, et RDFs. En conséquence, ils doivent être définis dans la partie préfix de l'espace de nommage comme suit

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
```

Et pour définir l'espace de nommage utilisé pour exprimer les données de notre ontologie qui doivent être mappées pour identifier les variables dans la requête

En ajoute le préfixe `PREFIX ns: <http://localhost:8080#> ;` où le ns est le nom de notre espace de nommage et http://localhost:8080# est le chemin d'ontologie utilisé.

En raison de extraire les éléments impactés dans la phase suivant. Le but de requête est d'avoir comme réponse une liste des éléments directement liée à un

élément donné. Par la suit, nous donnons la syntaxe des requête utilisées dans notre proposition.

Par exemple :

```
SELECT DISTINCT ?Req
WHERE {?Req rdf:type ns:requirement}
```

Dans cette requête, le prédicat '`rdf:type`' et le objet '`requirement`' participant aux triplets est recherché pour la ressource '`?Req`' tous les sujets de ces triples.

De même pour avoir tous les blocs d'un système

```
SELECT DISTINCT ?B
WHERE {?B rdf:type ns:block}
```

Et pour les actions dans activité

```
SELECT DISTINCT ?Act
WHERE {?Act rdf:type ns:action}
```

Tandis qui pour les relations dans ce type de requête il fallait juste définir le prédicat et dans ce cas-là est le type de relation qui liée entre le sujet et l'objet, les triplets cherchés sont tous les ressources qui participent à une des extrémités de cette relation.

Par exemple pour la relation « derived » qui lie entre deux exigences :

```
SELECT DISTINCT ?req1
WHERE {?req1 ns:derived ?req2}
order by desc(?req1)
```

La clause facultative `order by` est pour organiser la réponse selon le variable '`?req1`'.

```
SELECT DISTINCT ?req
WHERE {?A ns:verifiedby ?req}
order by desc(?req)
```

Pour savoir par quelle activité une exigence seront vérifié. Nous pouvons également distinguer entre les exigences par leur type car chaque exigence est caractérisée par son stéréotype. A l'aide le clause " OPTIONAL", par exemple la requête suivante permet avoir les exigences fonctionnelles.

```
SELECT ?req ?reqstereo ?reqid?reqtext
Where { ?req rdf:type ns:requirement.
Optional{?req ns:stereotype "fonctionnel _requirement"}}
```

Cette clause permettre de localiser une exigence via son Id ou son texte

```
SELECT ?req ?reqid?reqtext
Where { ?req rdf:type ns:requirement.
Optional {?req ns:id ?reqid}.
Optional{?req ns:text ?reqtext}.
```

De la même manière, nous localisent les blocs via leur stéréotype pour distinguer les blocs qui présentent les sous-systèmes d'un système.

```
SELECT ?B ?blocstereo
Where { ?B rdf:type ns:block.
Optional{?B ns:stereotype "sous_system"}}
```

Comme mentionné précédemment le SPARQL permet non seulement d'avoir la relation directe, par contre si nous voulons modifier une exigence liée à plusieurs autres exigences et satisfaite par un bloc, qui est lui-même décomposée en plusieurs blocs. Les éléments impactés par cette modification ne sont pas seulement les éléments directement liés mais, il existe une relation transitive indirecte. Pour extraire de tel relations, nous utilisons le langage de raisonnement SWRL

Les règles SWRL raisonnent sur les individus OWL, principalement en termes de classes et de propriétés OWL. Dans notre cas la règle SWRL exprimant que si un élément "A" du modèle ayant une relation "R1" avec un autre élément du modèle "B", et cet élément "B" ayant un relation "R2"avec un élément "C" de même modèle. Alors "A" a un relation "Impact" avec "C". Pour les éléments "A", "B", et "C" sont des concepts dans OWL, soit des classe ou sous-classe OWL, et pour les relations entre eux peuvent être exprimées à l'aide des propriétés OWL.

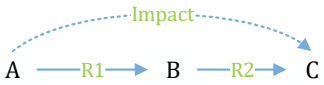


Figure IV.8 : Exemple avec relations déduites

La règle dans SWRL sera alors :

$$R1(A,B), R2(B,C) \rightarrow Impact(A,C)$$

À la fin de cette étape, nous avons acquis un outil qui nous permet de limiter les éléments liés directement ou indirectement à un élément particulier. Donc, si nous voulons changer cet élément, tous les éléments qui lui sont associés sont considérés comme éléments potentiellement impactés

IV.4.3 Degré d'impact

Notre objectif est de créer un outil qui aide à la décision en cas de modification, autrement dit mesure le degré d'impact de tous les éléments impactés. Pour se faire, nous prenons le résultat de la phase précédente et par la suite nous calculons la similarité sémantique entre l'élément à modifier et l'ensemble des éléments impactés d'un part et la modification proposé et le même ensemble d'une autre part. Ce calcul consiste à déterminer dans quelle mesure le sens des deux textes est similaire. Plus le score est élevé, plus la signification des deux phrases est similaire. Par conséquence, la relation est inverse entre le score de similarité et l'impact de changement car lorsque le score de similitude diminue, l'impact du changement augmente.

Pour comparer deux textes, il faut d'abord attribuer une similarité à des paires de mots. Nous proposons de mesurer la similarité sémantique en utilisant WordNet pour déterminer la similarité entre les mots.

Afin d'utiliser WordNet, nous transformons, d'abord les deux textes T1 et T2 en leur représentation de sac de mots, respectivement S1 et S2.

$$S1 = \{mots\ de\ T1\}$$
$$S2 = \{mots\ de\ T2\}$$

Nous utilisons le WordNet pour mesurer la similarité entre chaque paire de mots de S1 et S2. Pour obtenir à la fin un ensemble de score de similarité, en suit en regroupe cet ensemble en deux clusters : le premier C1 contient les scores de similarité supérieure ou égal 0.5 et le deuxième C2 contient les scores de similarité inférieur à 0.5. Selon le plus long cluster, nous déterminons si les deux textes sont similaires ou pas. C'est-à-dire si C1 est plus long que C2, cela signifie que les mots similaires sont supérieurs dans ce cas, le score de similarité est le maximum score de C1. Et vice versa. Les étapes de la procédure sont définies dans l'algorithme ci-dessous

```
Algorithm calculer le score de similarité sémantique
 // Entrée: paire de textes
  //sortie: score de similarité
Dédut
   S1 = (w1, w2 ,w3,…,wn);
   S2 = (w1, w2 ,w3,…,wm);
   Msim tableau de (len(S1)*len(S2))
   Pour chaque mot d'une S1
    Pour chaque mot d'un S2
        Msim←SimilaritéWordNet (S1(wi),S2(wj)) ;
        j←j+1
    fin;
    i←i+1;
   fin;
   Si (len(S1)=1) or (len(S2)=1) Alors Score ← max(Msim)
    sinon début
              pour i de 1 à len(S1)*len(S2)
                Si  Msim[i] ≥ 0.5 Alors début
                                        clusterSup[j] ← Msim[i];
                                         j←j+1;
                                       fin
                sinon  début
                          clusterInf[k] ← Msim[i] ;
                          k←k+1 ;
                        fin ;
                fin ;
              fin ;
          fin ;
   fin ;
   Si len(clusterSup)> len(clusterInf) Alors Score ← max(clusterSup)
      Sinon Score ← max(clusterInf) ;
Fin.
```

Comme résultat nous obtenons une liste des éléments impactés et leur degré de similarité avec l'ancien texte et avec le nouveau texte d'élément à modifier. Pour évaluer l'impact estimé, nous calculons la différence entre l'ancien et le nouveau score et nous le présentons comme une valeur entre 0 et 100. C'est la valeur d"impact.

IV.5 Conclusion

Dans ce chapitre, après avoir présenté certaines caractéristiques de la spécification de modélisation SysML, notamment les diagrammes d'exigences, de définition de bloc, et d'activité, nous avons donné une conception globale de notre approche qui se décompose en trois étapes commençant par proposer une formalisation OWL pour l'exploiter dans la transformation de la spécification SysML en une ontologie. La deuxième étape est consacrée à la mise en place des différentes requêtes et règles qui nous permettra d'analyser les différentes ressources de l'ontologie et dans la dernière étape nous avons proposé une démarche pour calculer la similarité sémantique entre les éléments pour permettre de mesurer l'impact des changements.

Nous consacrons le chapitre suivant à la réalisation et l'implémentation des différentes procédures de notre approche, ainsi qu'à un cas d'étude pour évaluer les résultats obtenus.

Chapitre V

Implémentation et étude de cas

I.1 Introduction

Afin d'outiller la méthodologie proposée, nous avons mis en place un outil permettant premièrement le passage automatique du modèle SysML à une présentation ontologique. Deuxièmement, l'utilisateur a la possibilité d'explore l'ontologie via une interface graphique associée à un ensemble de requêtes SPARQL prédéfinies. Et finalement, en cas de modification limité les éléments susceptibles d'être affectés par ce changement et déterminant leur degré d'impact. Par la suite présenter une expérimentation d'approche détaillé dans le chapitre précédent via l'évaluation de l'application développée

Pour ce faire, nous décrivons d'abord l'environnement de développement, puis nous présentons l'interface d'application conçue. Puis illustré l'utilisation de notre outil développé par une étude de cas.

V.2 Implémentation

Dans cette partie, nous décrivons d'abord l'environnement de développement et les outils utilisés avec les principales scripts développées. Ensuite, nous décrirons les différentes fenêtres de l'application tout en détaillant leurs fonctionnalités.

V.2.1 Environnement de développement

Pour implémenter notre approche nous avons choisi d'utiliser les de environnements suivants :

V.2.1.1 Polarsys

Polarsys est un projet de développement de logiciel open source, qui fournier un ensemble d'outils de développement de systèmes critiques et intégrés, de la spécification initiale à la mise en œuvre, puis à la vérification et à la validation. Polarsys héberge les composants de modélisation Eclipse tels que MDT Papyrus, Acceleo et plusieurs autres plateformes développement tels que le Sirius, EMF Compare, ATL et EMF Store.

Nous avons utilisé l'IDE Polarsys version 0.8 pour sa rapidité, sa facilité d'utilisation. De plus il comprend les deux outils, celui de la modélisation SysML, Papyrus. Et de la transformation M2T, Acceleo.[24]

V.2.1.2 Netbeans (Java)

NetBeans est un projet open source ayant un succès et une base d'utilisateur très large. Sun Microsystems a fondé le projet open source NetBeans en Juin 2000 et continue d'être le sponsor principal du projet. NetBeans supporter une variété langage y compris Java qui est un langage de P.O.O qui contribue à la fiabilité des logiciels et elle facilite la réutilisation de code existant grâce à sa portabilité en utilisant la JVM.

Nous avons utilisé l'environnement NetBeans version 8.2. C'est un environnement de développement et un outil pour les programmeurs pour écrire, compiler, déboguer et déployer des programmes. [7]

V.2.2 Les modules d'implémentions

Tout comme la conception nous avons segmenté notre implémentation en trois parties principales

V.2.2.1 La Transformation

Avant de parler de l'implémentation des règles de transformation d'un model SysML vers une forme ontologique, il faut d'abord créer ce dernier pour cela nous avons utilisé l'outils Papyrus.

Papyrus s'appuie sur la plate-forme Eclipse basées sur Polarsys Papyrus est une solution pour la modélisation SysML et UML. Il constitue la base de l'adoption et de l'utilisation de l'ingénierie dirigée par les modèles (MDE) et de l'ingénierie des systèmes basés sur les modèles (MBSE). [11]

Parmi les intérêts du choisi papyrus est qu'il se compose de plusieurs éditeurs principalement graphiques un d'eux concerne le SysML. Cet éditeur permet la visualisation simultanée de plusieurs diagrammes d'un modèle SysML donné, chaque diagramme pouvant exprimer un point de vue différent, dans notre cas nous avons besoin de Exigences, définition de bloc, activités. Ou plus le méta-modèle SysML est intégré dans EMF de POLARSYS qui comprennent aussi PAPYRUS

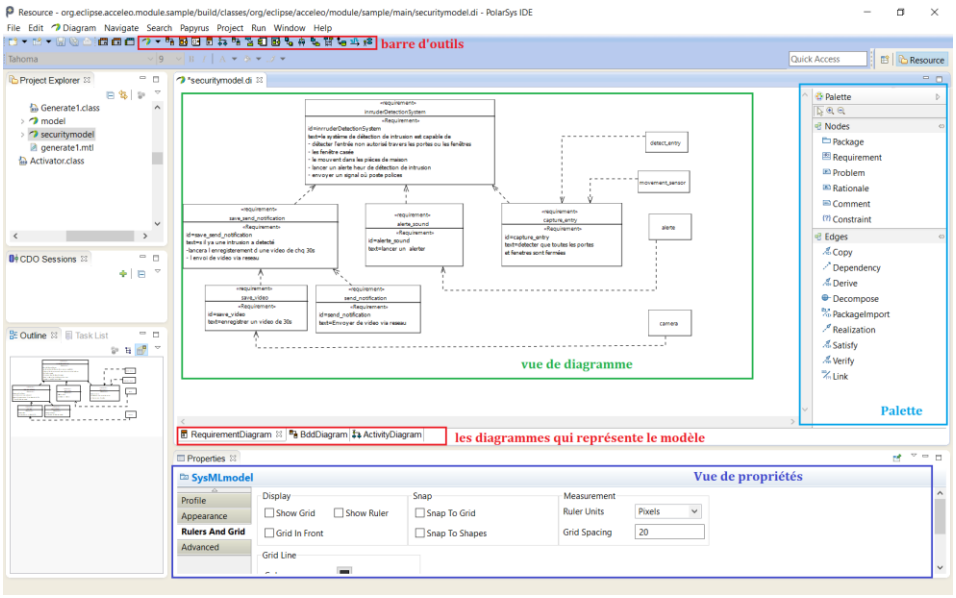


Figure V.1 : Composition d'un éditeur de diagramme

Nous avons utilisé la version 0.10.2 dans l'implémentation de notre approche. Donc, tout d'abord, nous devrions modaliser une solution en langage SysML avec un seul modèle contenant les trois diagrammes requis. Nous expliquerons plus en détail cette phase avec un exemple concret avec une étude de cas plus tard.

La transformation M2T est simplement extraire des informations sur les éléments de modèle SysML pour construire une autre structure suivant des règles pré déterminer. Pour se faire nous avons utilisé comme solution Acceleo.

"Acceleo est une technologie basée sur des modèles incluant des outils de création permettant de créer des générateurs de code personnalisés. Il permet de produire automatiquement tout type de code source à partir de toute source de données disponible au format EMF" [28]. Grâce à sa grande capacité de personnalisation, il nous offre le meilleur choix pour générer un format OWL à partir d'un modèle SysML. La fig V.2 représente un aperçu de processus de transformation.

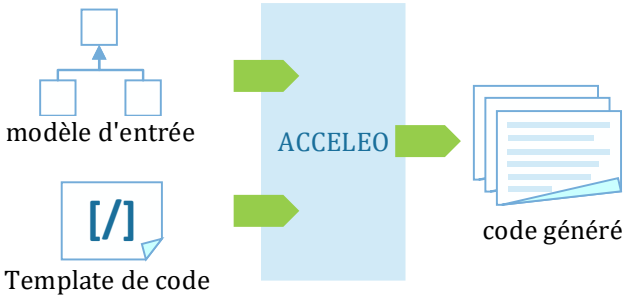


Figure V.2 : Vue d'ensemble Acceleo

Où les *modèle d'entrée* est le modèle implémenté via Papyrus. Et le *Template de code* est la structure utilisée dans Acceleo pour générer directement le code, ça peut être un ou plusieurs dans le même module Acceleo

Dans notre implémentation nous avons utilisé la version 3.4.2. Tous d'abord il faut configurer les métamodèles utilisés dans la transformation. Et comme dans papyrus le métamodèle SysML est construit en tant que profil intégré dans l'UML. Nous avons utilisé les deux comme suit

```
[comment encoding = UTF-8 /]
[module generate1( 'http://www.eclipse.org/uml2/4.0.0/UML',
                   'http://www.eclipse.org/papyrus/0.7.0/SysML')]
```

Le Template est la fonction principale d'un module Acceleo. Il a comme paramètre d'entrée un modèle ou une partie de celui-ci sur lequel les règles de transformation doivent être exécuté et il retourne des informations sur les éléments de ce modèle. Dans notre implémentation nous avons utilisés trois (03) Templates La première pour l'exigence, la deuxième pour l'activité, et le troisième pour les Block.

```
[template public generateElement(m:Requirement)]
[comment @main/]
[file ('Reqt.txt', true)]

    [comment]
        <!-- Individual_REQT     -->
    [/comment]

[/file]
[/template]

[template public generateElement(a:Activity)]
[comment @main/]
[file ('AD.txt', true)]

    [comment]
        <!--  Individual_ACTIVITY -->
    [/comment]

[/file]
[/template]

[template public generateElement(c:Block)]
[comment @main/]
[file ('Bdd.txt', true)]

    [comment]
        <!--  Individual_BDD -->
    [/comment]

[/file]
[/template]
```

Figure V.3 : Vue d'ensemble Templates

Il faut noter que chaque Template a son propre fichier de sortie. Donc le résultat de transformation est trois fichier texte. Où chaque un d'eux contient une partie de ABox concerne un type d'élément.

L'étape suivant est la mise en œuvre de règles de transformation dans chaque Template. Dans le première Template pour récupérer les détaille des exigences noua avons appliqués la règle suivante :

```
[comment]transforme un exigence[/comment]
 <!-- http://localhost:8080#[m.id/] -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#[m.id/]">
        <rdf:type rdf:resource="http://localhost:8080#Requirement"/>
        <id>[m.id/]</id>
        <Text>[m.text/]</Text>

    </owl:NamedIndividual>
<!-- ------------------------------------------------------------------->
```

Et pour parcourir la collection d'exigences afin d'obtenir les relations qui y sont participées, l'ensemble des règles seront comme suit :

```
[comment]transforme a composition relation[/comment]
[for(r:Requirement|m.master)]
<Decompose rdf:resource="http://localhost:8080#[r.id/]"/>
[/for]
[comment]transforme a derive relation[/comment]
[for(r:Requirement|m.derived)]
<Derived rdf:resource="http://localhost:8080#[r.id/]"/>
[/for]
[comment]transforme an verifiedBy relation[/comment]
[for(r3:Class|m.verifiedBy)]
<Verifiedby rdf:resource="http://localhost:8080#[r3.name/]"/>
 [/for]
[comment]transforme an satisfiedBy relation[/comment]
[for(b:Class|m.satisfiedBy)]
<Satisfiedby rdf:resource="http://localhost:8080#[b.name/]"/>
[/for]
[comment]transforme an refinedby relation[/comment]
[for(r4:Requirement|m.refinedBy)]
<Refinedby rdf:resource="http://localhost:8080#[r4.id/]"/>
[/for]
[comment]transforme an traceto relation[/comment]
[for(r5:Requirement|m.tracedTo)]
 <TracedTo rdf:resource="http://localhost:8080#[r5.id/]"/>
[/for]
```

Dans la deuxième partie noua avons appliqués trois règles de transformation. Un pour la transformation des activités

```
<!-- http://localhost:8080#[a.name/] -->
<owl:NamedIndividual rdf:about="http://localhost:8080#[a.name/]">
<rdf:type rdf:resource="http://localhost:8080#diagram_activity"/>
</owl:NamedIndividual>
```

L'autre pour l'ensembles des actions

```
[for(p:OpaqueAction | a.node)]
<!-- http://localhost:8080#[p.name/] -->
<owl:NamedIndividual rdf:about="http://localhost:8080#[p.name/]">
       <rdf:type rdf:resource="http://localhost:8080#action"/>
</owl:NamedIndividual>
[/for]
```

Et la dernière pour la transformation de relation de contenance qu'associe chaque action à l'activité à laquelle elle appartient

```
[comment]transforme les actions Contenu dans le diagramme d'activity [/comment]
[for(p:OpaqueAction | a.node)]
<contain rdf:resource="http://localhost:8080#[p.name/]"/>
[/for]
```

La troisième Template sert à transformer les blocs et les relations qui les combinent.

Ce qui concerne les blocs la règle de transformation est :

```
<!-- rdf:about="http://localhost:8080#[c.base_Class.name/]" -->
<owl:NamedIndividual rdf:about="http://localhost:8080#[c.base_Class.name/]">
    <rdf:type rdf:resource="http://localhost:8080#block"/>
</owl:NamedIndividual>
```

Dans ce qui suit la transformation de toutes les relations de diagramme de définition de block mentionné dans la partie conception.

```
[comment]transforme a composition relation[/comment]
[for(p: uml::Property | c.base_Class.ownedAttribute)]
<composition rdf:resource="http://localhost:8080#[p.name/]"/>
[/for]
[comment]transforme a generalization relation[/comment]
[for(p: uml::Property | c.base_Class.generalization)]
<generalization  rdf:resource="http://localhost:8080#[p.name/]"/>
[/for]
[comment]transforme a aggregation relation[/comment]
[for(p: uml::Property | c.base_Class.attribute.aggregation)]
<aggregation rdf:resource="http://localhost:8080#[p.name/]"/>
[/for]
[comment]transforme a association relation[/comment]
[for(p: uml::Property | c.base_Class.attribute.association)]
<association rdf:resource="http://localhost:8080#[p.name/]"/>
[/for]
[comment]transforme a relation between datatype and block[/comment]
[for(p: uml::Property | c.base_Class.attribute.datatype)]
<is_value_of rdf:resource="http://localhost:8080#[p.name/]"/>
[/for]
[comment]transforme a relation between block reference and block[/comment]
[for(p: uml::Property | c.base_Class.elementImport.relatedElement.eAnnotations.references)]
<is_reference_of rdf:resource="http://localhost:8080#[p.name/]"/>
[/for]
[comment]transforme a relation between block property and block[/comment]
[for(p: uml::Property | c.base_Class.collaborationUse.roleBinding)]
<is_property_of rdf:resource="http://localhost:8080#[p.name/]"/>
[/for]
[comment]transforme a relation between parts of block[/comment]
[for(p: uml::Property | c.base_Class.collaborationUse.name)]
<is_part_of rdf:resource="http://localhost:8080#[p.name/]"/>
[/for]
```

Finalement, il suffit de spécifier le modèle sur lequel les règles d'Acceleo seront appliquées pour lancer le générateur Acceleo. Et comme résultat nous obtenons trois fichiers texte qui constituent ensemble l'Abox, nous le concaténons avec la partie Tbox pour avoir notre ontologie dans sa forme finale.

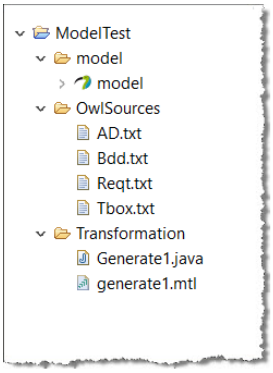


Figure IV.4 : Aperçu du résultat attendu

De plus, Eclipse nous permet d'exporter notre projet dans des fichiers JAR afin de l'importer ensuite à partir d'un autre environnement netbeand.

V.2.2.2 Analyse sémantique

Dans la partie analyse sémantique, notre implémentation consiste à stocker et à récupérer les données de résultat de la requête SPARQL dans une interface interactive, créé avec le langage Java en utilisant les API de Jena. Où, après avoir sélectionné le mot-clé de la requête, il sera analysé pour afficher le résultat.

Jena est un framework Java pour la création d'applications Web sémantique comprenant une API RDF lisant et écrivant des fichiers RDF au format RDF / XML, N3 et N-Triples, un moteur de requête SPARQL de l'API OWL en mémoire et à stockage persistant. [5]

Pour la mise en œuvre des requêtes SPARQL illustrés dans le chapitre IV, nous avons utilisés l'apache.jana où l'apache.jena.query est le package principal . En plus "d'autres packages contiennent diverses parties du système (moteur d'exécution, analyseurs, tests, etc.), nous mentionnons le plus important.

- *Query* : une classe qui représente la requête de l'application. C'est un conteneur pour tous les détails de la requête. Les objets de la classe *Query* sont créés via des méthodes de *QueryFactory* qui fournissent un accès à des analyseurs différents.
- *QueryExecutionFactory* : un endroit pour obtenir des instances de l'exécution d'une requête.
- *QuerySolution* : une solution unique à la requête.
- *ResultSet* : toutes les *QuerySolutions*. " [4]

Les étapes de base pour faire une requête *SELECTE*. Après introduis notre ontologie en tant que modèle, une requête est créée à partir d'une chaîne de caractères utilisant la *QueryFactory*. La requête et le modèle owl pouvait être interrogés sont ensuite transmis à *QueryExecutionFactory* pour produire une instance de l'exécution de la requête. Les résultats sont traités dans une boucle et finalement l'exécution de la requête est fermée.

Ce qui concerne l'implémentation de la règle de raisonnement sur l'ontologie mentionné dans la partie conception nous avons utilisés l'api Jena, et la syntaxe d'elle sera comme suit :

$$(?A \quad ?R1 \quad ?B), (?B \quad ?R2 \quad ?C) \rightarrow (?A \quad Impact \quad ?C)$$

V.2.2.3 Impact de changement

Afin d'implémenter la mesure d'impact de changement, comme nous l'avons déjà dit précédemment nous choisissons de calculer la similarité sémantique entre les mots de textes via le Wordnet. Pour cela nous utilisons le WS4j.

"WS4J (WordNet Similarity for Java) fournit une API Java pure pour plusieurs algorithmes publiés de similarité sémantique / similarité". Nous utilisons l'algorithme WuP pour calculer la similarité entre les mots qui retourne des valeurs comprises entre [0,1].

V.3 présentation de l'application

L'application que nous avons développée, basée sur les solutions proposées dans la conception présentée dans le chapitre précédent, consiste sur un interface homme machine nous décrivons les principales fenêtres ci-dessous. L'interface principale de notre application est présentée dans la figure suivante :

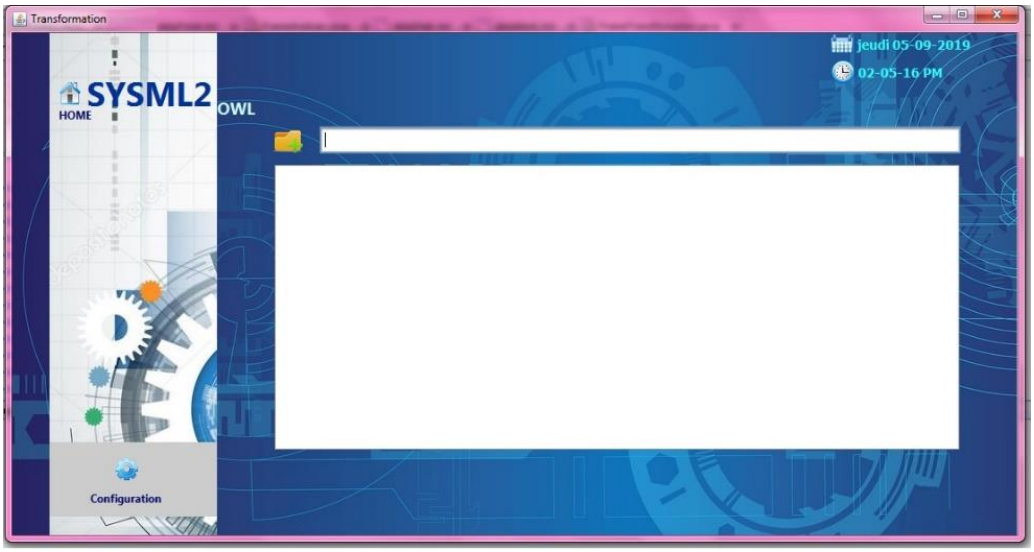


Figure V.5 : Interface principale d'application

L'utilisateur entre le chemin vers son modèle SysML. Exécuté et consulté la transformation en clique sur le bouton transformation, et enregistre le résultat en format OWL. Cette interface permet également à l'utilisateur d'ignorer l'étape de transformation ou de passer à l'étape d'analyse représentée dans cette interface.

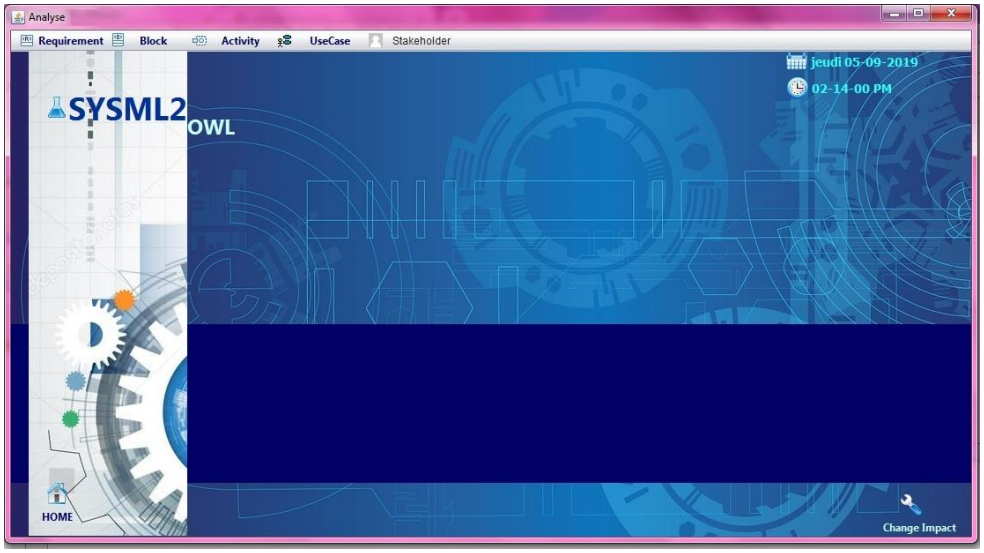


Figure V.6 : L'interface d'analyse de système

Elle compose d'un menu horizontal contient 4 catégories et des sous menus vertical contient le détaille de chaque un d'eux, pour donner à l'utilisateur le choix de spécifie une catégorie ou un composant d'elle. Et selon le choix d'utilisateur, il peut avoir pour chaque élément de système par quels éléments sont liés et par quelles relations. En conséquence l'utilisateur peut déterminer les éléments impactés en cas de modification, et pour avoir le degré d'impact en passe à l'interface suivant:

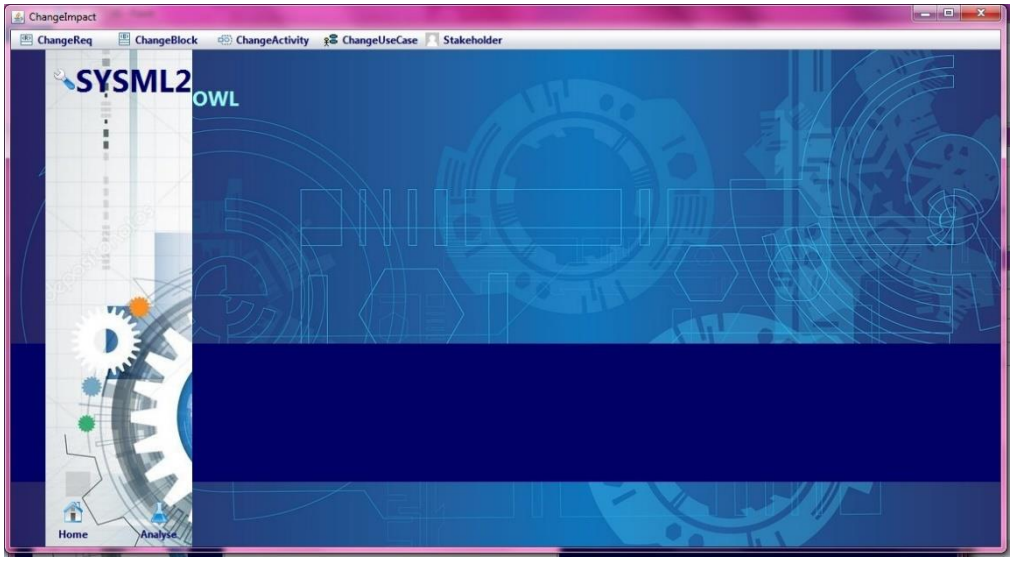


Figure V.7 : l'interface de mesure l'impact de changement

Cette interface nous permet de calculer l'impact de chaque changement, grâce à la possibilité de sélectionner n'importe quel élément du système et à la possibilité de proposer un amendement, en résultat une liste des éléments affectés par ce changement sera affichée avec le calcul de la similarité entre ce qui existait et ce qui est proposé, puis mesurez l'impact de la modification pour chacun de ces éléments impactés.

V.4 Etude de cas

Pour une illustration optimale des étapes de notre méthodologie, nous présentons une étude de cas du système d'airbag d'une voiture.

V.4.1 Présentation de système

Les systèmes embarqués automobiles peuvent être divisés en domaine fonctionnels « centrés sur le véhicule », tels que le contrôle du groupe motopropulseur, le contrôle du châssis et les systèmes de sécurité active ou passive, et en domaine fonctionnels « centrés sur le passager » où multimédia / télématique, corps / confort et sécurité humaine. Le domaine de la carrosserie contient des fonctions intégrées dans un véhicule qui concernent les essuie-glaces, les phares, les portes, les vitres, les sièges, les airbags et les rétroviseurs.

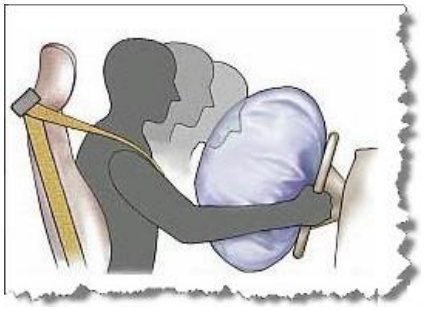


Figure V.8 : Système d'Airbag

Le système de contrôle des Airbags est un élément de sécurité capable d'atténuer l'impact d'une éventuelle collision et de protéger la vie des passagers. Couplé à la ceinture de sécurité, l'airbag protège le conducteur et tout passager contre les effets d'un choc frontal sur le véhicule. Le système d'airbag n'est pas activé en cas de choc latéral, de choc arrière ou de renversement. Il est activé uniquement si l'impact se produit dans un angle maximal de 30 degrés par rapport au mouvement avant. Des capteurs et des actionneurs sont intégrés à l'unité de contrôle du système. Les capteurs détectent un net ralentissement et identifient le moment où la vitesse varie d'au moins 20 km / h en peu de temps, comme lors d'une collision. L'actionneur vise à déterminer le moment du déclenchement de l'airbag. L'unité de commande envoie un signal électrique à l'allumeur chargé de gonfler l'airbag. Au sein de l'allumeur, des substances telles que les nitrates et la guanidine ammoniacale réagissent et explosent instantanément. La réaction chimique génère suffisamment d'azote pour remplir le sac en seulement 30 millisecondes. L'airbag commence à se vider lors de l'impact avec le corps. L'airbag contient un gaz chimique générateur à l'état solide. Ce gaz est stocké dans une chambre en métal. Lorsque l'unité de commande reçoit le signal du capteur, un courant électrique est appliqué au sac.

Tel système avancé dont l'objectif est de contrôler l'accès au véhicule, doivent respecter de strictes contraintes en temps réel.

Modélisation des exigences d'airbag

Selon les caractéristiques mentionnées précédemment et l'analyser de domaine détaillé qu'est a effectué, les principales exigences requises pour tel système est modélisées comme suit :

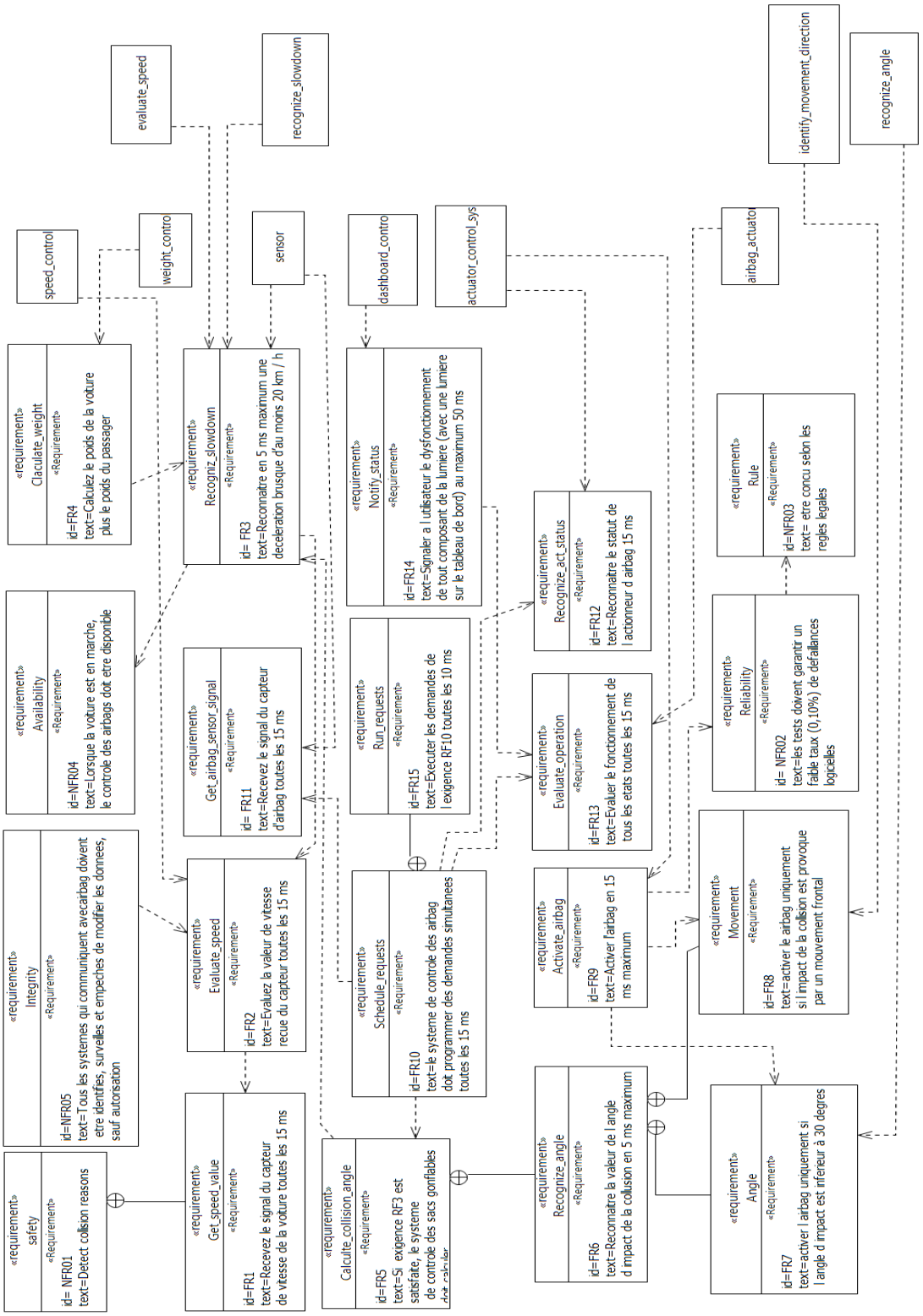


Figure V.9 : Diagramme des exigences SysML

Modélisation des Bloc d'airbag

Pour modéliser les composants impliqués dans le système d'airbag. Les fonctions spécifiées sont implémentées pour des communications sécurisées avec d'autres composants et avec des capteurs / actionneurs locaux. Le diagramme BDD SysML est illustré à la *Figure V.10*.

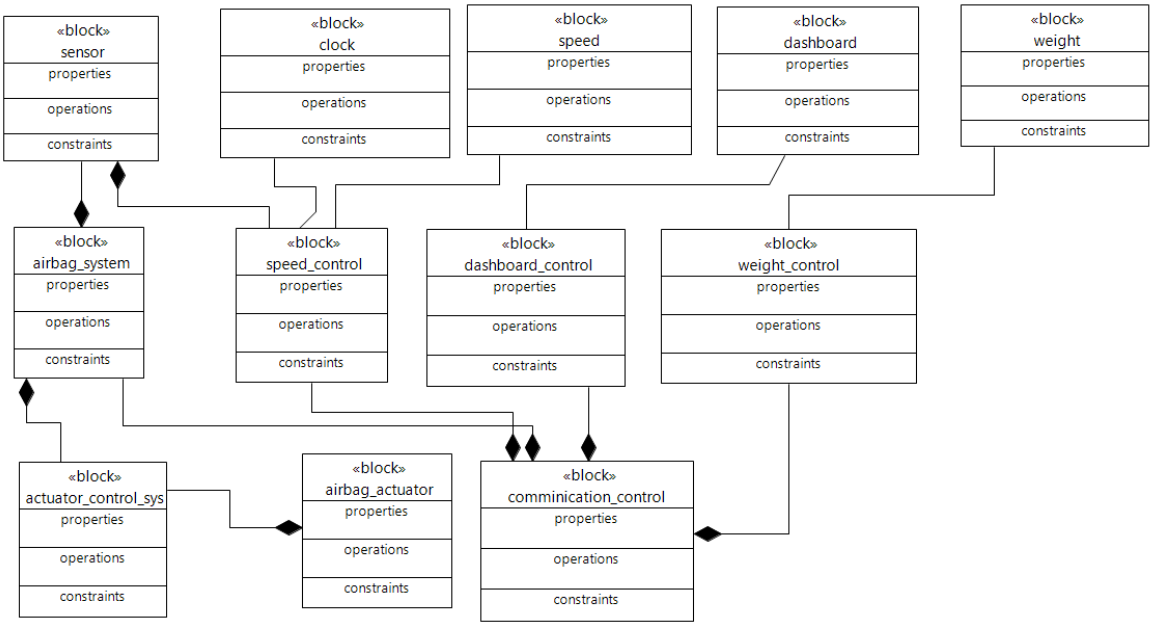


Figure V.10 : Diagramme BDD

Diagramme d'activité SysML d'airbag

Nous représentons le diagramme d'activité principal du système de contrôle des sacs gonflables à la *Figure V.11*. Ce diagramme présente l'activité d'activation de l'airbag

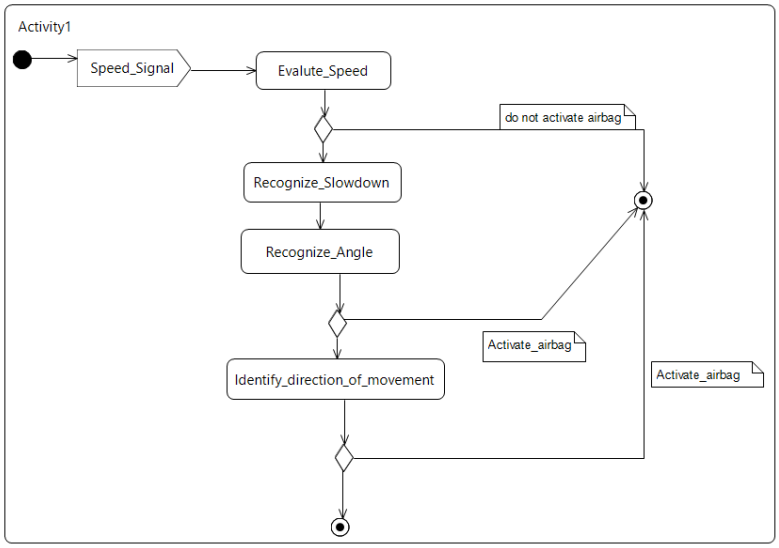


Figure V.11 : Diagramme d'activité SysML

V.4.2 Transformation de modèle

Après avoir modélisé et enregistré ce système en SysML via l'outil PolarSys, la première étape consiste à introduire ce modèle dans l'application en utilisant l'interface présentée dans la fig V.12. On commence par choisir le chemin du modèle du système, on obtient alors une fenêtre qui s'ouvre, on sélectionne le modèle.

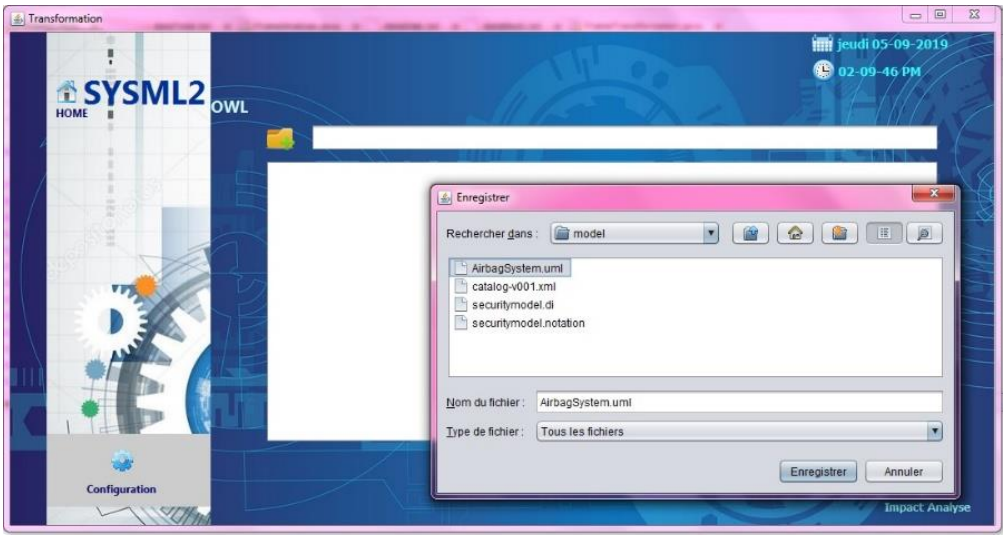
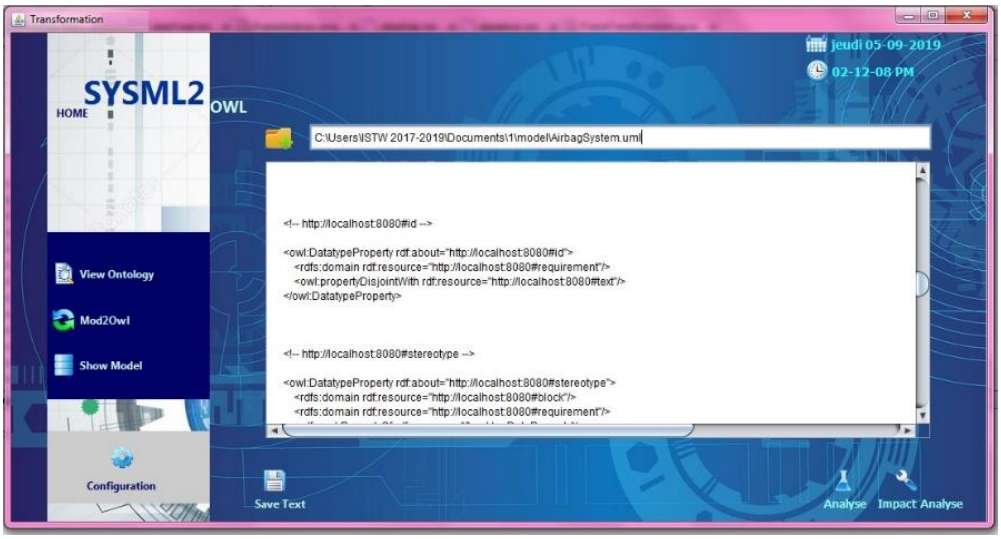


Figure V.12 : Interface graphique de sélection le modèle

On déclenche la transformation du modèle en cliquant sur le bouton "Mod2Owl". Le résultat de transformation sera affiché en cliquant sur le bouton "View ontology". On peut l'enregistrer pour le réutiliser par lasuite. Pour accéder aux deux types d'analyse, cliquer sur le choix souhaité.



FigureV.13 : Interface graphique de Transformation de modèle

V.4.3 Analyse de système

Pour l'interface d'analyse, on choisit un type de diagramme parmi les types apparaissant dans le menu principal, ceci est montré par la fig V.14.

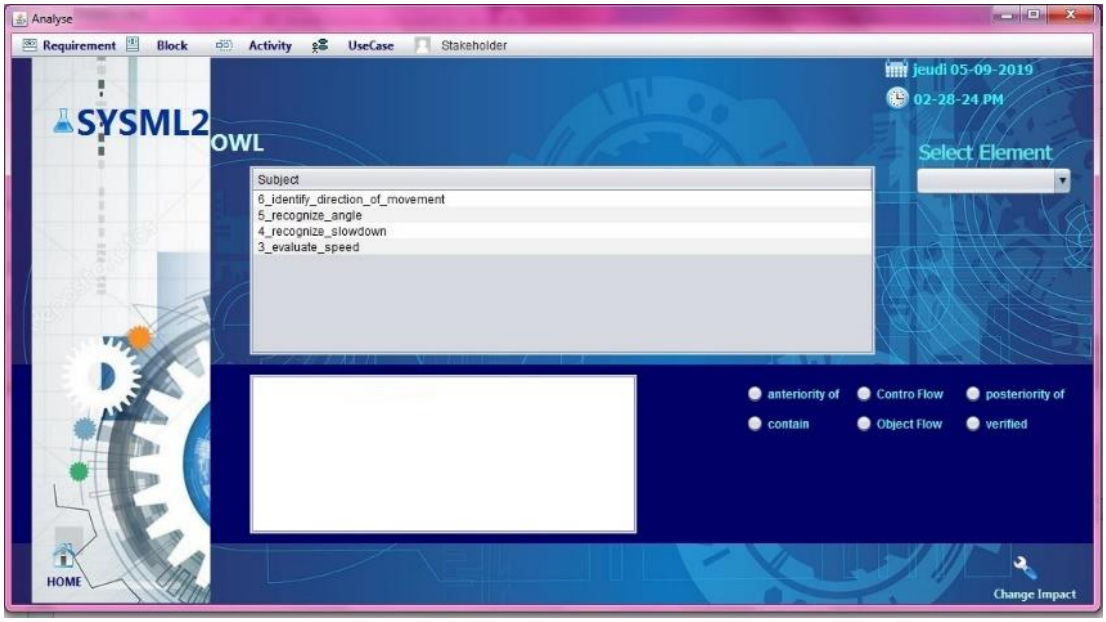


Figure V.14 : Interface graphique de sélection Analyse de éléments d'un diagramme de Système

Dans notre cas, on a choisi le diagramme d'activité pour savoir toutes les actions, il suffit de cliquer sur l'item "action" dans le sous-menu "Activity", pour plus de précision il suffit de sélectionner l'élément désiré "Select Element" et la relation souhaite comme présentée dans la fig V.15.

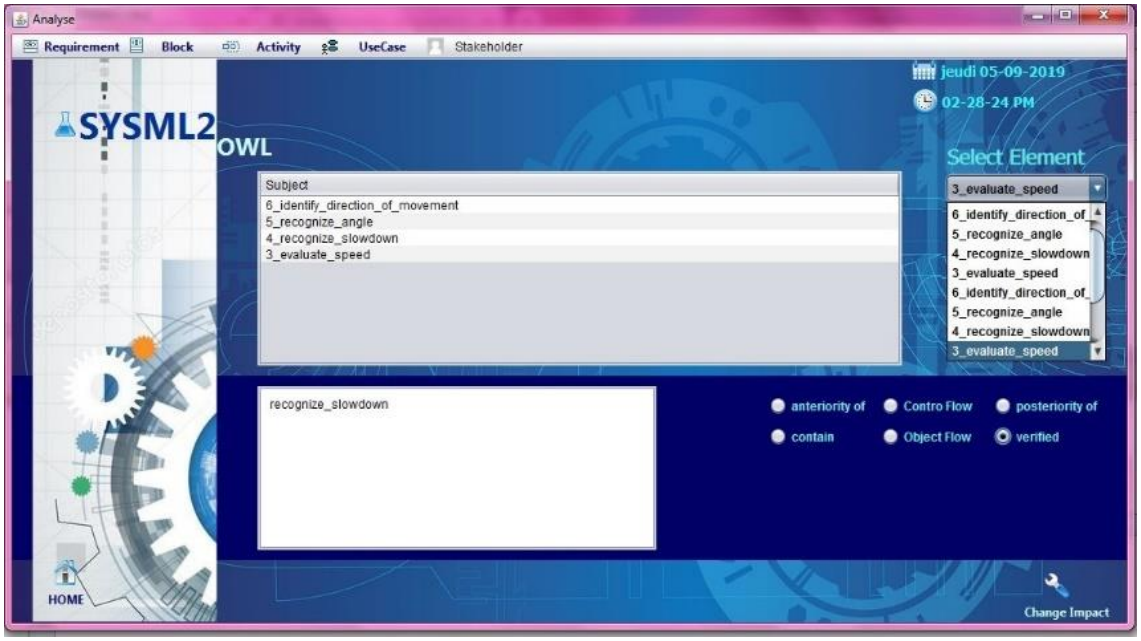


Figure V.15 : Interface graphique d'analyse de par élément / relation

V.4.4 Analyse de Impact de changement

Dans la phase analyse de l'impact de changement, on distingue deux types de changement : soit suppression et dans ce cas, on sélectionne le type de diagramme ensuit l'élément à supprimer.

Dans cet exemple, on a choisi de supprimer le bloc " angel ", dans ce cas qui aspectent d'être impacte sont mapper dans le graphe suivant :

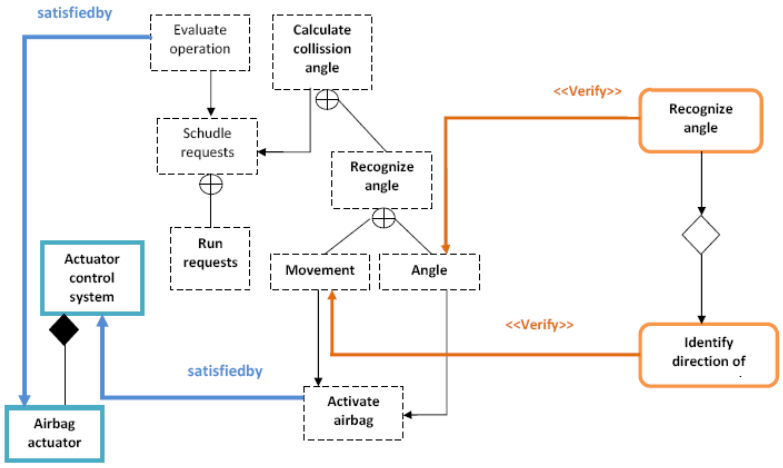


Figure V.16 : mappage des éléments impact par la suppression d'élément « angel »

Deux types d'éléments impactés s'affichent : le premier, les éléments directement impactés car il existe une relation directe entre " angel ", et les éléments indirectement impactés.

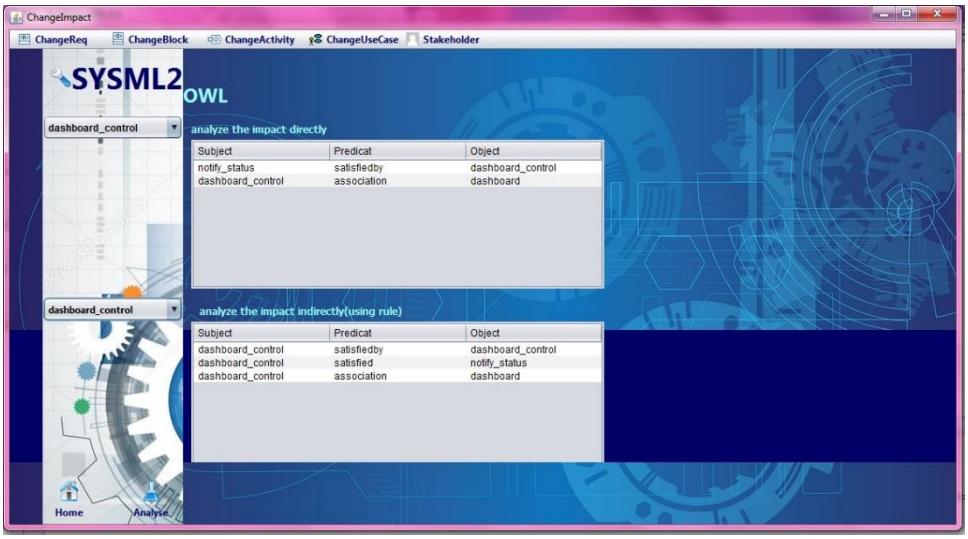


Figure V.17 : Interface graphique d'analyse d'impact de suppression d'un élément

Concernant la suppression nous proposent de supprimer le block « angel » dans ce cas qui aspectent d'être impacte sont mapper dans le graphe suivant

On peut choisir de modifier un élément. Dans ce cas, une zone de texte sera affichée pour introduire le nouveau texte. Dans l'exemple motionné dans la fig V.17 on a choisi de modifier le texte de l'exigence "angle", une fois le nouveau texte est introduit une liste des éléments impactés sera affichée avec le degré d'impact de chaque élément.

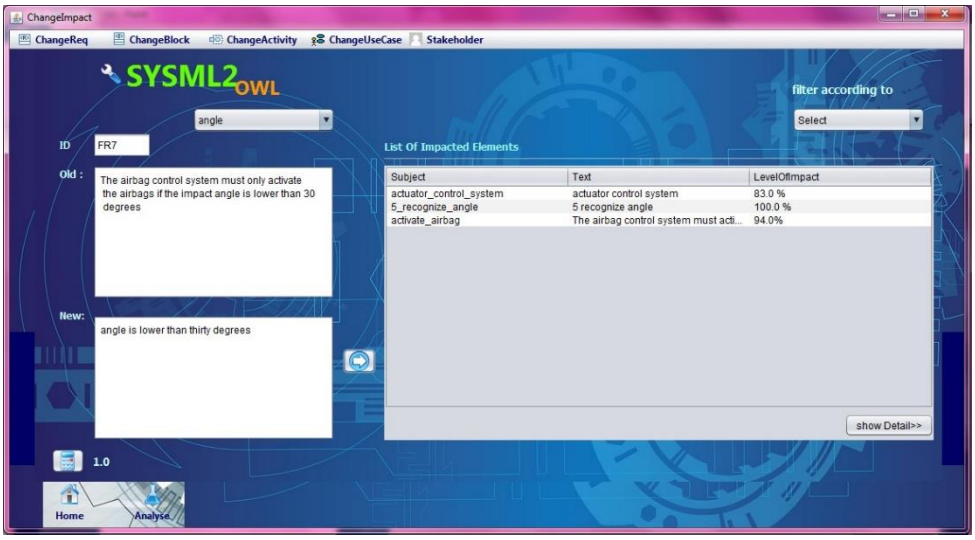


Figure V.18: Interface graphique d'analyse d'impact de modification d'un élément sans détaille

Il est possible d'avoir la liste des éléments impactés avec plus de détaille, en affichant les similarités sémantiques entre l'ancien texte et les éléments impactés et les similarités entre le nouveau texte et les mêmes éléments.

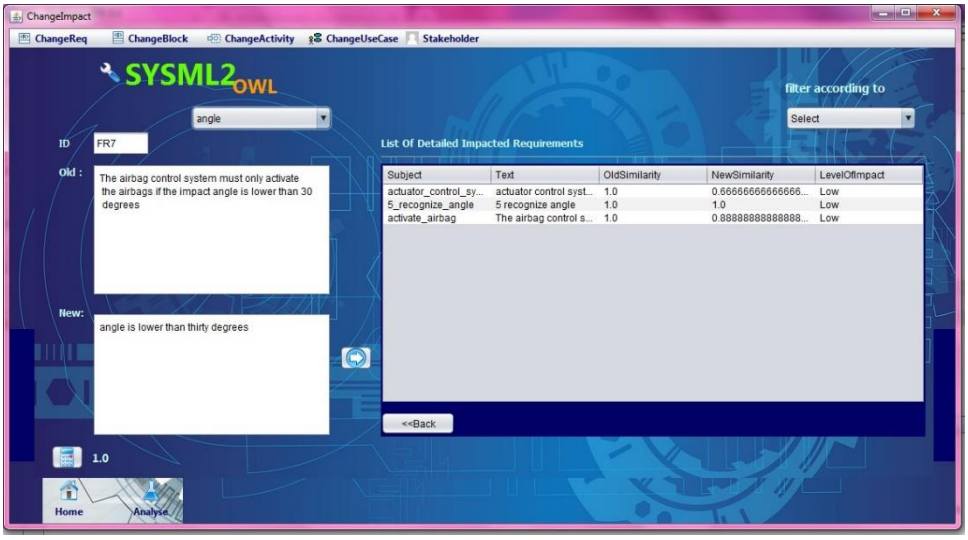


Figure V.19: Interface graphique d'analyse d'impact de modification d'un élément détaillée

Comme il est possible de filtrer les éléments impactés en fonction de leur bloc, de leur type,…etc.

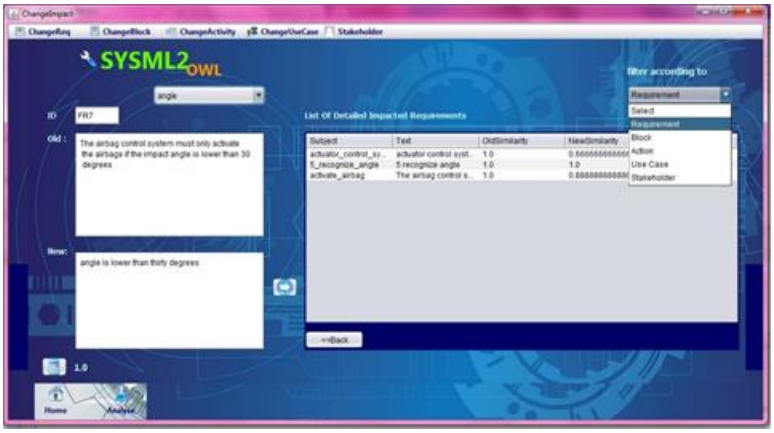


Figure V.20 : Interface graphique d'analyse d'impact de modification filtrée par exigence

V.4.5 Test et Résultat

Comme test d'évaluation de notre proposition pour calculer l'impact de changement nous démontrons dans ce qui suit les résultats obtenus en cas de change le texte d'exigence "Safety"

Texte : the airbag control system must detect collusion reasons.

Et le nouveau texte sera

Texte : angle is lower than thirty degrees

Les éléments qui aspectent d'être impacte sont mapper dans le graphe suivant

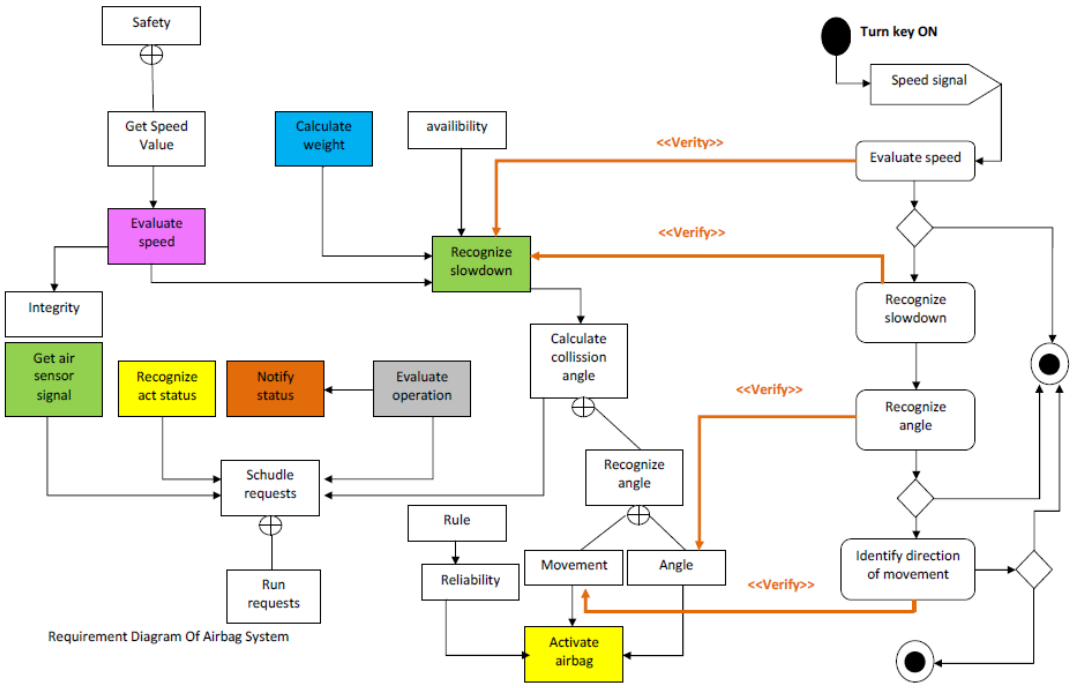


Figure V.21 : mappage des éléments impact par le changement d'élément « Safety »

Tableau V.1 : résultat obtenu

Elément	Niveau d'impact
recognize_slowdown	94%
run_requests	94%
recognize_angle	100%
integrity	90%
calculate_collusion_angel	90%
movement	97%
evaluate_speed	94%
activate_airbag	94%
schedule_requests	94%
calculate_weight	100%
angel	100%
get_speed_value	90%

Concernant la suppression nous proposent de supprimer le block « angel » dans ce cas qui aspectent d'être impacte sont mapper dans le graphe suivant

V.5 Conclusion

Après avoir présenté l'environnement de travail et montrer les outils que nous avons utilisé et les étapes d'implémentation de notre outil, nous avons valorisé noter méthodologie via une étude de cas suivant plusieurs axes, la facilité de transformation, l'efficacité d'analyse, la simplicité de proposer un changement et déterminer son effet immédiat sur le reste du système.

Conclusion Général

Le langage de modélisation de systèmes SysML a rencontré un large succès en tant que notation graphique standard pour le domaine de l'ingénierie des systèmes. Cependant SysML a une fondation sémantique permettant un raisonnement et une analyse limités. Car est basée sur la formalisation partielle d'UML. Ce qui constitue un obstacle important à l'obtention d'une grande confiance dans la validité des conclusions tirées de ces analyses. OWL, en revanche, a été développé dès le début sur des principes de logique formels, et fournit par conséquent un support puissant pour la vérification de la cohérence et de la satisfiabilité, et l'extraction des implications.

La combinaison entre la notation graphique attrayante de SysML et du raisonnement formel d'OWL nous avais motivée de proposer une approche dans le contexte de l'ingénierie dirigée par les modèles pour passer de modélisation SysML vers une formalisation OWL. Pour pouvoir raisonner sur un système, et par la suit implémenter un mécanisme pour déterminer les éléments de modèle potentiellement impactés par une modification de n'importe quel élément de système.

Afin d'aboutir à notre objectif, nous avons à travers une recherche bibliographique vue les principales notions de modélisation basée sur modèle, et la représentation des connaissances notamment les ontologies. Ensuite les différentes approches pour mesurée la similarité textuelle. Finalement, nous avons proposé la méthode d'analyse d'impact de changements basés sur SysML et les ontologies

Notre approche consiste à représenter SysML dans OWL d'une manière automatique grâce a un mécanisme simple a manipuler avec et qui extraire les connaissances d'une façon optimal. Le préanalysée des ontologies à l'aide de requêtes SPARQL afin de générer des résumés pour éléments de système et les relations entre eux, ce mappage nous a aidées par la suit pour limiter les éléments impactés en cas dans changement, finalement mesuré sémantiquement la similarité entre les éléments impactés et l'élément à modifie d'un part et les éléments impactés et la modification proposé d'un autre part, en résultat, augmenter la précision de mesure l'impact de changement.

Dans l'état actuel de nos recherches, de nombreuses études sont encore nécessaires pour atteindre une génération complète de représentation de connaissances de SysML par une ontologie en complétant les règles de transformation pour inclure tous le métamodèle SysML, au plus alimenter cette ontologie par des règles de raisonnement pour qu'elle soit plus complète. Développement d'ontologies de disciplines qui étendent les concepts de base, par exemple, électrique, mécanique, vérification, etc. ce qui améliorent nécessairement les résultats obtenus par le calcul de similarité sémantique.

En effet, suite à notre étude l'objectif principal a été atteint, nous proposons l'utilisation, éventuellement, l'hybridation avec les futures études complémentaires de ce travail.

Bibliographie et Webographie

[1] Abdellaoui. N : Application de l'approche d'ingénierie dirigée par les modèles pour le développement des applications mobiles (Interfaces graphiques), Mémoire de Master, Université Abou Bakr Belkaid Tlemcen, Algérie, 2016.

[2] Abid. A : Approche basée sur l'Ingénierie Système pour la Conception Intégrée des Systèmes Manufacturiers Reconfigurables, Thèse de doctorat, Université Paris-Saclay, France, 17Jun2019

[3] Anthony Legendre : Ingénierie système et Sûreté de fonctionnement : Méthodologie de synchronisation des modèles d'architecture système et d'analyse de risques, Thèse de doctorat, Université Paris-Saclay, FRance, 2017

[4] ARQ - API Application, Le Club Developpez.com, https://web-semantique.developpez.com/tutoriels/jena/arq/api-application/, 10/08/2019, 6h13.

[5] Bansal. R & Chawla. S: Semantic Web Tool-For Efficient Retrieval of Links and Required Information, International Journal of Innovative Technology and Exploring Engineering (IJITEE), V3, 2013

[6] Beldjerdi. M & Dougdag. S : Vers une vérification distribuée des diagrammes d'activités SysML, mémoire de master, Université Yahia Fères Médéa, Algérie, 2016

[7] Bienvenue à NetBeans et www.netbeans.org : Apache NetBeans, https://netbeans.org/index_fr.html, 31/07/2019,10h29.

[8] Cardoso.J & Miguel Pinto.A : The web ontology language (OWL) and its Applications, IGI Global,2015

[9] David ten Hove, Change Management within SysML Requirements Models, Mémoire de Master, Université de Twente ()Pays-Bas), 2010.

[10] Dragos.D, "Contribution à la modélisation d'un système interactif d'aide à la conduite d'un procédé industriel", Thèse de doctorat, Université Henri Poincaré - Nancy I, Français,2010.

[11] Eclipse Papyrus: eclipse foundation https://projects.eclipse.org/projects/modeling.mdt.papyrus, 02/08/2019,08h05.

[12] Friedenthal. S & Moore. A & Steiner. R: OMG Systems Modeling Language (OMG SysML) Tutorial, Object Management Group,2009.

[13] Functional and Non-functional Requirements-Specification and Types: AltexSoft, https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/, 10/03/2019,9h28

[14] Feldmann, S., Kernschmidt, K., & Vogel-Heuser, B, "Combining a SysML-based modeling approach and semantic technologies for analyzing change influences in manufacturing plant models", Procedia Cirp 17, 2014.

[15] Goknil. A, Kurtev.I, Berg. K , "A Rule-Based Change Impact Analysis Approach in Software Architecture for Requirements Changes" , 2016

[16] HACINE GHERBI.A: Construction d'une Ontologie pour le WEB Sémantique, Mémoire de Magister, Université Ferhat Abbes Sétif1 Algérie, 2014

[17] http://homes.cs.washington.edu/~billhowe/cs410/: W3C, https://www.w3.org/TR/owl-features/, 05/05/2019,1h56

[18] Introduction aux technologies du Web Sémantique : eurecom, http://www.eurecom.fr/fr/course/WebSem-2019Spring,06/04/2019,6h08

[19] Jarraya, Yosr and Debbabi, Mourad and Bentahar, Jamal, On the Meaning of SysML Activity Diagrams, 2009

[20] Karakostas.B & Zorgios.Y: Engineering Service Oriented Systems-A Model Driven Approach, IGI Publiching Hershey New York, 2008

[21] Le Moigne, J. L : La théorie du système général Théorie de la modélisation (2e éd.). Paris, Presses universitaires de France, 1977.

[22] Logique descriptive et OWL : Michel Gagnon, https://moodle.polymtl.ca/, 05/09/2019, 4h39

[23] Lond.D & Scott.Z: Model-Based Systems Engineering 2nd edition, Vitech, October 2011

[24] Matthew Hause, Francis Thom,Building Bridges Between Systems and Software with SysML and UML,INCOSE with permission,2008

[25] Modélisation des exigences avec SysML: Magazine.org, https://re-magazine.ireb.org/articles/modeling-requirements-with-sysml, 02/06/2019,16h55

[26] Negre.E , Comparaision de textes quelques approches ..., chahier du LAMSADE 338, 2013

[27] O'Connor.MJ & Knublauch.H & Tu. & Musen. M: Writing Rules for the Semantic Web using SWRL and Jess, 2019

[28] Overview: Acceleo, https://www.eclipse.org/acceleo/overview.html, 03/08/2019, 11h03.

[29] PolarSys Top Level Project Charter: eclipse foundation ,https://projects.eclipse.org/projects/polarsys/charter,31/07/2019,9h8.

[30] RDF Schema 1.1, W3C, https://www.w3.org/TR/2014/REC-rdf-schema-20140225/,14/04/2019,10h02

[31] Resource Description Framework, Champin.P A, https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/semweb/rdf.html?highlight=jena, 10/04/2019,9h31

[32] Roques.P : SysML par l'exemple -Un langage de modélisation pour systèmes complexes, Eyrolles, 2009

[33] SPARQL Query Language for RDF: W3C, https://www.w3.org/2001/sw/DataAccess/rq23/rq24.html, 13/04/2019, 5h56

[34] SysML : le diagramme d'exigence (requirement diagram), https://www.urbanisation-si.com/sysml-le-diagramme-d-exigence-requirement-diagram, 05-07-2019,9h26

[35] SysML: urbanisation-si, https://www.urbanisation-si.com/articles/sysml, 19/02/2019,12h11

[36] Systems Engineering, https://www.incose.org/systems-engineering, 18/02/2019,10h21

[37] Systems Engineering: Object Management Group, https://www.omg.org/hot-topics/syse.htm, 15/04/2019,11h37

[38] Vàcha. P, Texture Similarity Measure, WDS'05 Proceedings of Contributed Papers, Part I, 47–52, 2005.

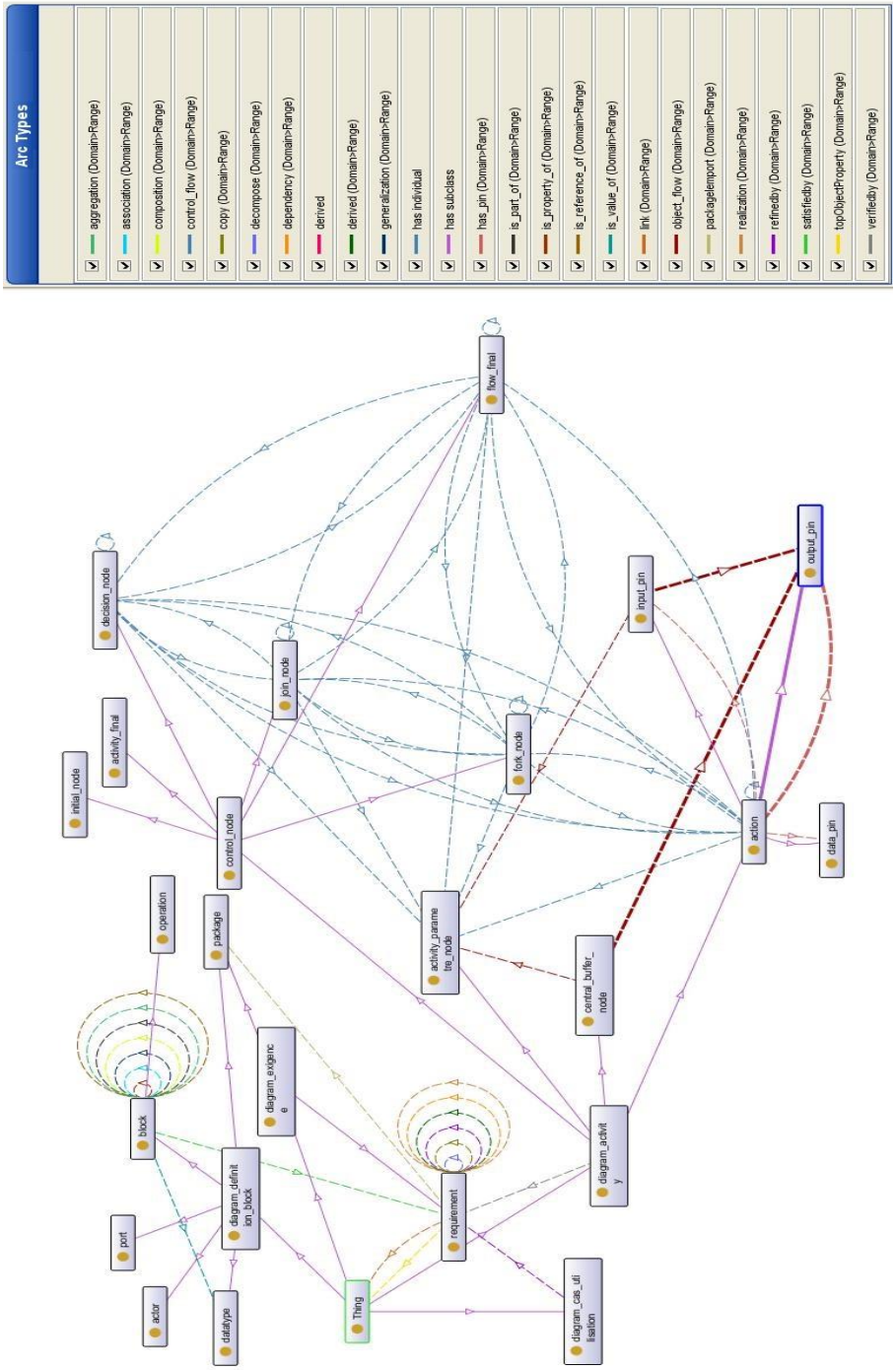
[39] Vasutiu Gestion, Gestion des changements et étude d'impact dans un système d'information réglementaire, Vasutiu, Ovidiu and Jouve, David and Amghar, Youssef

[40] W3C: OWL 2 Web Ontology LanguageProfiles, W3C, Editor's Draft, 20 April 2009

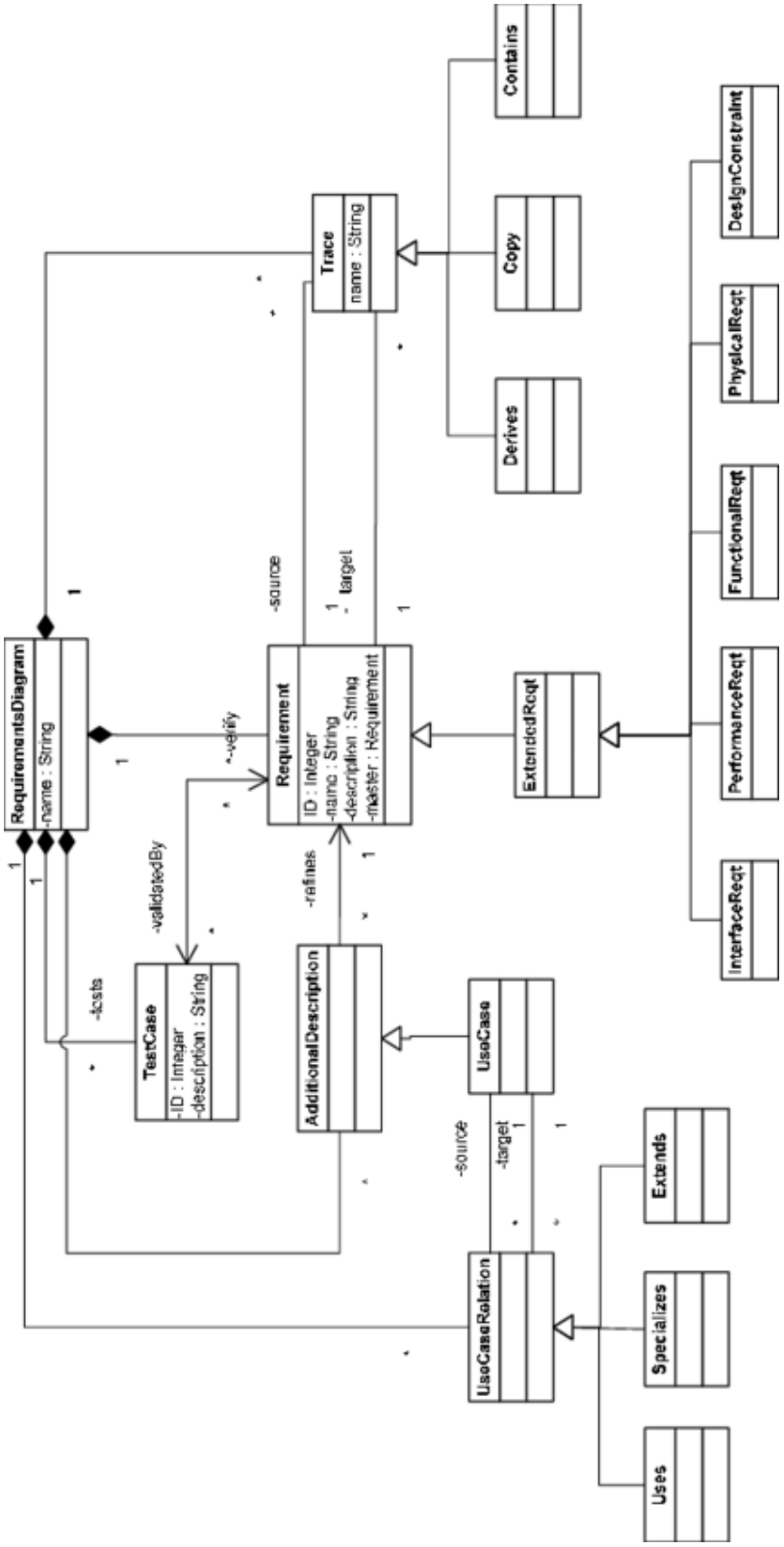
[41] www.ibm.com/products/software

Annexes

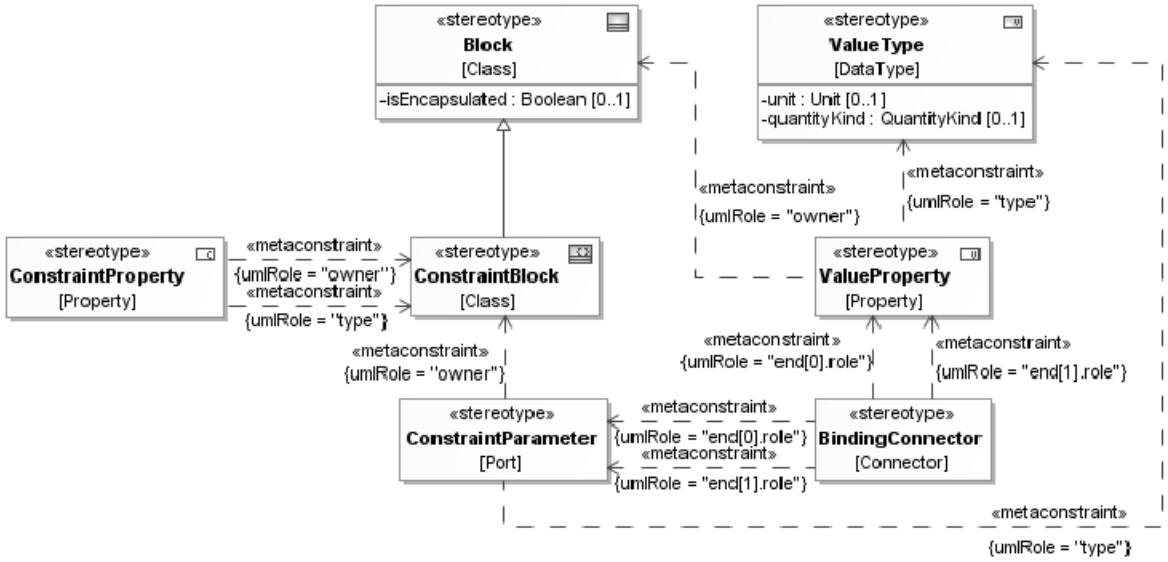
1. Tbox d'ontologie SysML (presenté via protégé)



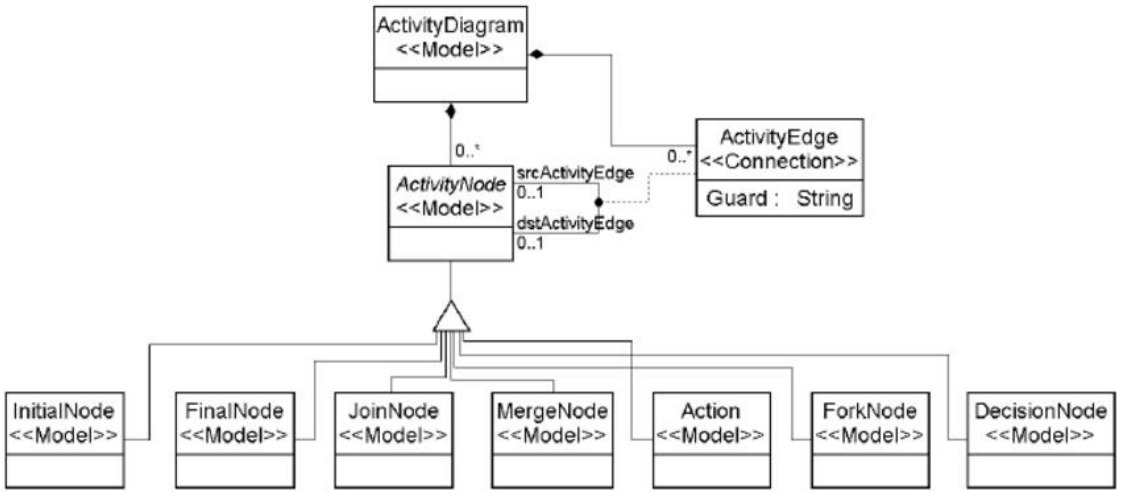
2. Le Métamodèle de diagramme d'exigence



3. Le Métamodèle de diagramme de définition de block



4. Le Métamodèle de diagramme d'activité



5. Présentation d' Abox de l'ontologie de système airbag

```
//////////////////////////////////////////////////////////////////////////
    // Individuals
//////////////////////////////////////////////////////////////////////////
    <!-- http://localhost:8080#Activate_airbag -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Activate_airbag">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <text>The airbag control system must activate the airbags in a
maximum of 15 ms after collision impact.</text>
        <stereotype>Functional_Requirements</stereotype>
        <id>FR09</id>
        <satisfiedby
rdf:resource="http://localhost:8080#actuator_control_sys"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Angle -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Angle">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <id>FR07</id>
        <text>The airbag control system must only activate the airbags if
the impact angle is lower than 30 degrees.</text>
        <stereotype>Functional_Requirements</stereotype>
        <derived rdf:resource="http://localhost:8080#Activate_airbag"/>
        <verifiedby
rdf:resource="http://localhost:8080#Recognize_Angle_act"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Availability -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Availability">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Non_Functional_Requirements </stereotype>
        <text>When the car is in operation, the airbag control system must
be available 99,6 % of time.</text>
        <id>NFR04 </id>
        <derived rdf:resource="http://localhost:8080#Recogniz_slowdown"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Calculte_collision_angle -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Calculte_collision_angle">
```

```
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <text>If the requirement RF3 is satisfied, then the airbag control system must calculate the collision impact angle in exactly 5 ms.</text>
        <id>FR05</id>
        <stereotype>Functional_Requirements</stereotype>
        <decompose rdf:resource="http://localhost:8080#Recognize_angle"/>
        <derived rdf:resource="http://localhost:8080#Schedule_requests"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Claculate_weight -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Claculate_weight">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Functional_Requirements</stereotype>
        <id>FR04</id>
        <text>When the car is in operation, the airbag control system must calculate the car weight plus the passenger weight in a maximum of 5 s
</text>
        <derived rdf:resource="http://localhost:8080#Recogniz_slowdown"/>
        <satisfiedby rdf:resource="http://localhost:8080#weight_control"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Evaluate_operation -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Evaluate_operation">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <id>FR13.</id>
        <text>The airbag control system must evaluate the functioning of all components every 15 ms</text>
        <stereotype>Functional_Requirements </stereotype>
        <derived rdf:resource="http://localhost:8080#Notify_status"/>
        <derived rdf:resource="http://localhost:8080#Schedule_requests"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Evaluate_speed -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Evaluate_speed">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Functional_Requirements</stereotype>
        <text>The airbag control system must evaluate the speed value received from sensor every 15 ms.</text>
        <id>FR02</id>
```

```
        <derived rdf:resource="http://localhost:8080#Get_speed_value"/>
        <derived rdf:resource="http://localhost:8080#Integrity"/>
        <is_authoredBy
rdf:resource="http://localhost:8080#ingenieur_automobile"/>
        <satisfiedby rdf:resource="http://localhost:8080#speed_control"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Evalute_Speed_act -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Evalute_Speed_act">
        <rdf:type rdf:resource="http://localhost:8080#action"/>
        <antériorité_de
rdf:resource="http://localhost:8080#Recognize_Slowdown_act"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Get_airbag_sensor_signal -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Get_airbag_sensor_signal">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Functional_Requirements</stereotype>
        <id>FR11</id>
        <text>The airbag control system must get the airbag sensor signal
every 15 ms.</text>
        <decompose rdf:resource="http://localhost:8080#Schedule_requests"/>
        <is_authoredBy
rdf:resource="http://localhost:8080#ingenieur_automobile"/>
        <satisfiedby rdf:resource="http://localhost:8080#sensor"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Get_speed_value -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Get_speed_value">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <text>The airbag control system must get the speed sensor signal of
car every 15 ms.</text>
        <id>FR01</id>
        <stereotype>Functional_Requirements</stereotype>
        <derived rdf:resource="http://localhost:8080#Evaluate_speed"/>
        <refinedby rdf:resource="http://localhost:8080#TestCase0"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Identify_Direction_Movement_act -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Identify_Direction_Movement_act">
```

```
        <rdf:type rdf:resource="http://localhost:8080#action"/>
        <postériorité_de
rdf:resource="http://localhost:8080#Recognize_Angle_act"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Integrity -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Integrity">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <id>NFR05</id>
        <text>The airbag control system must be righteous,all systems which
communicate with the airbag system must be identified, monitored and
blocked from changing data, unless they are authorized.</text>
        <stereotype>Non_Functional_Requirements</stereotype>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Movement -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Movement">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Functional_Requirements</stereotype>
        <id>FR08</id>
        <text>The airbag control system must only activate the airbags if
the collision impact is at frontal movement.</text>
        <derived rdf:resource="http://localhost:8080#Activate_airbag"/>
        <verifiedby
rdf:resource="http://localhost:8080#Identify_Direction_Movement_act"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Notify_status -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Notify_status">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <text>The airbag control system must notify to user the malfunction
of any light component (with a light on the dashboard) in a maximum of 50
ms.</text>
        <id>FR14</id>
        <stereotype>Functional_Requirements</stereotype>
        <satisfiedby
rdf:resource="http://localhost:8080#dashboard_control"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Recogniz_slowdown -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Recogniz_slowdown">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
```

```
        <text>The airbag control system must recognize in a maximum of 5 ms
an abrupt deceleration of at least 20 km/h.</text>
        <stereotype>Functional_Requirements</stereotype>
        <id>FR03</id>
        <derived
rdf:resource="http://localhost:8080#Calculte_collision_angle"/>
        <derived rdf:resource="http://localhost:8080#Claculate_weight"/>
        <verifiedby
rdf:resource="http://localhost:8080#Evalute_Speed_act"/>
        <verifiedby
rdf:resource="http://localhost:8080#Recognize_Slowdown_act"/>
        <refinedby rdf:resource="http://localhost:8080#TestCase2"/>
        <is_authoredBy
rdf:resource="http://localhost:8080#ingenieur_automobile"/>
        <satisfiedby rdf:resource="http://localhost:8080#sensor"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Recognize_Angle_act -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Recognize_Angle_act">
        <rdf:type rdf:resource="http://localhost:8080#action"/>
        <antériorité_de
rdf:resource="http://localhost:8080#Identify_Direction_Movement_act"/>
        <postériorité_de
rdf:resource="http://localhost:8080#Recognize_Slowdown_act"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Recognize_Slowdown_act -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Recognize_Slowdown_act">
        <rdf:type rdf:resource="http://localhost:8080#action"/>
        <postériorité_de
rdf:resource="http://localhost:8080#Evalute_Speed_act"/>
        <antériorité_de
rdf:resource="http://localhost:8080#Recognize_Angle_act"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Recognize_act_status -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Recognize_act_status">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <text>The airbag control system must recognize the airbag actuator
status every 5 ms.</text>
```

```
        <stereotype>Functional_Requirements</stereotype>
        <id>FR12</id>
        <derived rdf:resource="http://localhost:8080#Schedule_requests"/>
        <satisfiedby
rdf:resource="http://localhost:8080#actuator_control_sys"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Recognize_angle -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Recognize_angle">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Functional_Requirements</stereotype>
        <id>FR06</id>
        <text>The airbag control system must recognize the value of
collision impact angle in a maximum of 5 ms.</text>
        <decompose rdf:resource="http://localhost:8080#Angle"/>
        <decompose rdf:resource="http://localhost:8080#Movement"/>
        <refinedby rdf:resource="http://localhost:8080#TestCase1"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Reliability. -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Reliability.">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Non_Functional_Requirements</stereotype>
        <id>NFR02 </id>
        <text>The car must have reliability airbag system - tests must
guarantee low rate (0.10%) of software failures</text>
        <derived rdf:resource="http://localhost:8080#Activate_airbag"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Rule -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Rule">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Non_Functional_Requirements </stereotype>
        <text>The airbag control system must be designed according to legal
rules (omitted for review).</text>
        <id>NFR03 </id>
        <derived rdf:resource="http://localhost:8080#Reliability."/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Run_requests -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Run_requests">
```

```
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Functional_Requirements</stereotype>
        <id>FR15</id>
        <text>The airbag control system must run requests of requirement
RF10 every 10 ms.</text>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Schedule_requests -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#Schedule_requests">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Functional_Requirements</stereotype>
        <text>The airbag control system must schedule simultaneous requests
every 15 ms.</text>
        <id>FR10</id>
        <decompose rdf:resource="http://localhost:8080#Run_requests"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#Speed_Signal -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#Speed_Signal">
        <rdf:type
rdf:resource="http://localhost:8080#central_buffer_node"/>
        <object_flow
rdf:resource="http://localhost:8080#Evalute_Speed_act"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#TestCase0 -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#TestCase0">
        <rdf:type
rdf:resource="http://localhost:8080#diagram_cas_utilisation"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#TestCase1 -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#TestCase1">
        <rdf:type
rdf:resource="http://localhost:8080#diagram_cas_utilisation"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#TestCase2 -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#TestCase2">
        <rdf:type
rdf:resource="http://localhost:8080#diagram_cas_utilisation"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#act_ActivateAirbag -->
```

```
    <owl:NamedIndividual
rdf:about="http://localhost:8080#act_ActivateAirbag">
        <rdf:type rdf:resource="http://localhost:8080#flow_final"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#act_DoNotActivateAirbag -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#act_DoNotActivateAirbag">
        <rdf:type rdf:resource="http://localhost:8080#flow_final"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#activate_airbag_ActDia -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#activate_airbag_ActDia">
        <rdf:type rdf:resource="http://localhost:8080#diagram_activity"/>
        <contain rdf:resource="http://localhost:8080#Evalute_Speed_act"/>
        <contain
rdf:resource="http://localhost:8080#Identify_Direction_Movement_act"/>
        <contain rdf:resource="http://localhost:8080#Recognize_Angle_act"/>
        <contain
rdf:resource="http://localhost:8080#Recognize_Slowdown_act"/>
        <contain rdf:resource="http://localhost:8080#Speed_Signal"/>
        <contain rdf:resource="http://localhost:8080#act_ActivateAirbag"/>
        <contain
rdf:resource="http://localhost:8080#act_DoNotActivateAirbag"/>
        <contain rdf:resource="http://localhost:8080#test_angle"/>
        <contain rdf:resource="http://localhost:8080#test_movement"/>
        <contain rdf:resource="http://localhost:8080#test_speed"/>
        <contain rdf:resource="http://localhost:8080#turn_key_on"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#actuator_control_sys -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#actuator_control_sys">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
        <composition rdf:resource="http://localhost:8080#airbag_actuator"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#airbag_actuator -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#airbag_actuator">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
```

```
        <stereotype>Block</stereotype>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#aribag_system -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#aribag_system">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
        <composition
         rdf:resource="http://localhost:8080#actuator_control_sys"/>
        <composition rdf:resource="http://localhost:8080#sensor"/>
</owl:NamedIndividual>
    <!-- http://localhost:8080#clock -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#clock">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
        <association rdf:resource="http://localhost:8080#speed_control"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#comminication_control -->
    <owl:NamedIndividual
rdf:about="http://localhost:8080#comminication_control">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
<composition rdf:resource="http://localhost:8080#aribag_system"/>
        <composition
         rdf:resource="http://localhost:8080#dashboard_control"/>
        <composition rdf:resource="http://localhost:8080#speed_control"/>
        <composition rdf:resource="http://localhost:8080#weight_control"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#dachboard -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#dachboard">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
        <association
rdf:resource="http://localhost:8080#dashboard_control"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#dashboard_control -->
    <owl:NamedIndividual
      rdf:about="http://localhost:8080#dashboard_control">
```

```
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#ingenieur_automobile -->
    <owl:NamedIndividual
      rdf:about="http://localhost:8080#ingenieur_automobile">
        <rdf:type rdf:resource="http://localhost:8080#stakeholder"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#safety -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#safety">
        <rdf:type rdf:resource="http://localhost:8080#requirement"/>
        <stereotype>Non_Functional_Requirements </stereotype>
        <text>The airbag control system must detect collision
reasons.</text>
        <id>NFR01 </id>
        <decompose rdf:resource="http://localhost:8080#Get_speed_value"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#sensor -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#sensor">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
        <composition rdf:resource=" http://localhost:8080#speed_control"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#speed -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#speed">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
        <association rdf:resource="http://localhost:8080#speed_control"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#speed_control -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#speed_control">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#test_angle -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#test_angle">
```

```
        <rdf:type rdf:resource="http://localhost:8080#decision_node"/>
        <control_flow
rdf:resource="http://localhost:8080#Identify_Direction_Movement_act"/>
        <control_flow
rdf:resource="http://localhost:8080#Recognize_Angle_act"/>
        <control_flow
rdf:resource="http://localhost:8080#act_DoNotActivateAirbag"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#test_movement -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#test_movement">
        <rdf:type rdf:resource="http://localhost:8080#decision_node"/>
        <control_flow
rdf:resource="http://localhost:8080#Identify_Direction_Movement_act"/>
        <control_flow
rdf:resource="http://localhost:8080#act_ActivateAirbag"/>
        <control_flow
rdf:resource="http://localhost:8080#act_DoNotActivateAirbag"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#test_speed -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#test_speed">
        <rdf:type rdf:resource="http://localhost:8080#decision_node"/>
        <control_flow
rdf:resource="http://localhost:8080#Evalute_Speed_act"/>
        <control_flow
rdf:resource="http://localhost:8080#Recogniz_slowdown"/>
        <control_flow
rdf:resource="http://localhost:8080#act_DoNotActivateAirbag"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#turn_key_on -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#turn_key_on">
        <rdf:type rdf:resource="http://localhost:8080#initial_node"/>
        <object_flow rdf:resource="http://localhost:8080#Speed_Signal"/>
    </owl:NamedIndividual>
    <!-- http://localhost:8080#weight -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#weight">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
        <association rdf:resource="http://localhost:8080#weight_control"/>
    </owl:NamedIndividual>
```

```
    <!-- http://localhost:8080#weight_control -->
    <owl:NamedIndividual rdf:about="http://localhost:8080#weight_control">
        <rdf:type rdf:resource="http://localhost:8080#block"/>
        <stereotype>Block</stereotype>
        <composition
rdf:resource="http://localhost:8080#comminication_control"/>
    </owl:NamedIndividual>
```