République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Seddik BEN YAHIA-JIJEL

Faculté des Sciences Exactes et Informatique

Département d'informatique

Mémoire

De fin d'étude pour l'obtention

Du diplôme : Master de Recherche en Informatique

Option : *Système d'information & Aide à la Décision (SIAD)*

Thème

Modélisation des systèmes cyber-physiques : Cas d'un Pacemaker

Encadré par :

Mr. YAHIAOUI Abdelbaki

Réalisé par:

BELHENNACHE Asma

Promotion 2019/2020

« Remerciements »

Tout d'abord je remercie le Dieu tout puissant et miséricordieux, qui m'a donné la volonté, la force, la patience d'accomplir ce travail.

Je tiens aussi à exprimer mon sincère reconnaissance et mon profonde gratitude à mon encadreur Mr " YAHIAOUI ABDELBAKI " pour son soutien, sa disponibilité, ses orientations, ses précieux conseils et ses encouragements qui m'a permis d'élaborer ce travail dans des bonnes conditions.

Je voudrais aussi exprimer ma reconnaissance envers tous les membres du jury.

Sans oublier de présenter mon sincères remerciements à ma grand familles , qui ont étaient toujours de j'encourager durant mon parcours de mes études, ainsi que pour leurs aides, leurs compréhensions et leurs soutiens.

J'adresse mes remerciements les plus sincères à et mes amis, qui m'ont toujours encouragé au cours de la réalisation de ce mémoire.

Je souhaite adresser mes remerciements les plus sincères et ma reconnaissance au professeur "BOUAZIZ HAMIDA","LAHOULOU ITIDAL" et "BOUINAH MADIHA "

Enfin, je tiens à remercier aux personnes qui nous ont apporté leur aide et qui ont contribué d'une façon ou d'une autre à l'élaboration de ce mémoire.

« Dédicaces »

Tout l'amour et l'appréciation à

Mon très cher Père CHERIF (dieu ait pitié de lui), Ma tendre Mère zoubida,

Mon cher fiancé Riyadh ,mon grand père tayab,

Mes sœurs et frère Israa, Rokia ,Islem pour leurs encouragements

Permanents aussi sihem, jahida, salah .

mes tantes ,mes oncles avec

ses enfants et femmes

à Touts ma grands famille chacun avec son nom

.

A Mes très chères amies Radia ,fariel, Sara ,Hadjer, Wafa,Salima, Rania .

aussi mes camarades de promotion 2020.

Merci d'être toujours là pour moi

ASMA

Résumé

L’objectif visé par ce projet de fin étude est l étude et la modélisation des systèmes cyber-physiques.

Le cas d'étude retenu est le pacemaker qui est un dispositif électronique implanté dans le cœur d’un patient souffrant de bradycardie ou à l’inverse de tachycardie.

Les outils utilisés dans la phase de mise en œuvre sont : Dia et Atom3.

Mots clés : Systèmes Cyber-physiques, Modélisation, AToM3, **Dia,Mda, Idm,**

Modèle ,Méta_modèle, Méta_méta modèle

Table de matière

Table de matière

Liste de tableaux

Table des figures

Liste des abréviations

Introduction générale 12

1. Les systèmes cyber-physique 14

1.1 Les caractéristiques les plus importantes 14

1.1.1 Les caractéristiques de cps et domaine d'utilisation 16

1.1.2 Quelque exemples 17

1.2 La modélisation des systèmes cyber-physiques 19

1.2.1 Introduction. 19

1.2.2 Architecture Dirigée par les Modèles 19

1.2.3 Modèles formels des applications 19

1.2.4 Le langage AADL. 21

1.3 conclusion 21

2. La transformation de modèles 25

2.1 introduction 25

2.2 ingénierie dirigée par les modèles 25

2.2.1 Concepts fondamentaux 26

2.2.1.1 modèle 26

2.2.1.2 méta-modèle. 26

2.2.1.3 méta méta modèle 26

2.2.2 Architecture dirigée par les modèles 28

2.2.2.1 Principe du MDA 28

2.2.2.2 Différents modèles du MDA 29

2.2.2.3 Transformation de modèles MDA 30

2.3 Transformation de modèles 31

2.3.1 définition générales 31

2.3.1.1 transformations endogènes et exogènes 32
2.3.1.2 Transformation horizontales et verticales. 32
2.3.2 concepts de base et principe. 32
2.3.2.1 transformation de type modèles vers code. 33
2.3.2.2 transformation de type modèles vers modèles. 33
2.3.3 caractéristiques d'une transformation de modèles 33
2.3.4 outils et langages de transformation 34
2.3.5 classification des outils de transformation 36
2.3.6 la validation 36
2.4 conclusion 37
3. Etude de cas : Le Pacemaker 38
3.1 Introduction 38
3.2 Le cœur 40
3.3 Pacemaker 41
3.4 Spécification du cas d'étude 40
3.5 Définition du Système 41
3.6 Cycle de vie du système 42
3.7 Exigence de système 43
3.8 Comportements de pacemaker 47
3.9 Modélisation du pacemaker 50
3.10 Conclusion 52

4. Implémentation et études de cas 53
4.1 Introduction. 53
4.2 dia 53
4.3 AToM3 59
4.3.1 Présentation de l'outil AToM3 59
4.3.2 Exemples de cas du pacemaker dans AtoM3 62
4.4 Conclusion 74
Conclusion générale 75
Bibliographie 75

Liste des tableaux

3.1 - Mode opérationnel bradycardie d'un pacemaker.

Table de figures

1.1 Domaine d'utilisation du CPS …… 14

1.2 Composantes d'un système cyber physique …….. 15

1.3 Défférentes caractiristiques d'un CPS …… 16

1.4 Prototype d'équipement personnel de santé (Santos et al., 2015) ……. 18

1.5 Schéma de principe d'un système de transmission de puissance électrique …….. 18

1.6 L'approche MDA …… 20

1.7 Types et implantations de composants AADL …….. 22

1.8 Possibilités d'extension des composants …… 22

1.9 Syntaxe graphique des composants logiciels ……. 23

1.10 Syntaxe graphique des composants matériels ……. 24

2.1 Niveaux de modélisation ……. 27

2.2 Autre image pour préciser les niveaux de modalisation …… 27

2.3 Les principaux modèles ……... 29

2.4 Transformation des modèles MDA ……. 31

2.5 Concepts de base et principe des transformations de modèles …… 33

3.1 Schéma du cœur …… 34

3.2 Pacemaker de Boston Scientific …… 40

3.3 Le système global ……. 51

3.4 Processus PG …… 51

3.5 Le sous-système LEAD …… 51

3.6 Communication PG – DCM …….. 52

4.1 Quelques outils Dia …… 54

4.2 L'interface de dia …… 55

4.3 Organisation de la feuille de travail dans Dia …… 55

4.4 Fenêtre principale de dessin ……. 56

4.5 Les composants matériels d'un pacemaker ……. 57

4.6 Modélisation des composants matériel d'un pacemaker …… 57

4.7 Modélisation des composants matériel d'un pacemaker ……. 58
4.8 Diagramme représente le système global …….. 58
4.9 La modélisation du système globale dans dia …… 59
4.10 Interface d'AToM3 ……. 60
4.11 étape 1 pour créer un model dans atom3 …….. 60
4.12 étape 2 pour créer un model dans atom3 …… 61
4.13 étape 3 pour créer un modèle dans atom3 …….. 61
4.14 Atom3 avec Barr d'outils …… 61
4.15 contraintes temporelles1 ……. 62
4.16 diagramme1 de classe dans atom3 …….. 62
4.17 Le bouton 'Graphical_Appearance' dans l atom3 …….. 63
4.18 Designer l'état initial à partir de l'éditeur …… 63
4.19 Sauvegarde le dessin …… 64
4.20 Création d'état final1 …… 64
4.21 Génération du modèle1 …… 65
4.22 Suit de génération1 ……... 65
4.23 Le méta model de diagramme1 …… 66
4.24 Faire une relation entre l état initial et l'action …….. 66
4.25 Contraintes temporelles 2 …….. 66
4.26 Diagramme de classe de Contraintes temporelles 2 …….. 67
4.27 Editeur graphique de la classe initiale 2 …….. 67
4.28 Editeur graphique d'une action 2 …….. 68
4.29 Le méta modèle 2 obtenu …….. 68
4.30 Réalisation des relations entres les états2 ……. 69
4.31 Diagramme de Contraintes temporelles 3 …… 69
4.32 Le diagramme de classe 3 …… 70
4.33 Le méta model de modèle 3 …….. 70
4.34 Editeur de grammaires d'AToM3 …… 71

4.35 Editeur de règles d'AToM3 …… 72

4.36 Le méta modèle du modèle ……. 72

4.37 La création de la règle dans l'éditeur de règles d'AToM3 …… 73

4.38 La sauvegarde de la règle ……. 73

4.39 L'exécution de la règle ……. 74

4.40 Le résultat de l'exécution de la règle …….. 74

Liste des abréviations

CPS	Cyber-Physiquel System
NFS	National Science Fondation
PHD	Personnel Health device
IDO	Internet Des Objets
MBSE	Model Best Systèmes
IDM	Ingénierie Dirigée par les modèles
MDE	Model Driven Engineering
OMG	Object Management
MDA	Architecture Diriger par Model
QVT	Query Vieu Transformation
CIM	modèle indépendant de calcule
PDM	Modèle Plus Forme
PSM	Modèle Dépendent des Plates-Forme
OQL	Object Constraint Language
LHS	Left-Hand Side
RHS	Right-Hand Side
AToM3	A Tool For Mlti-formalism and Meta-Modelling
MSDL	Modelling, Similation and Design Lab

Un système cyber-physique est un système ou des éléments informatiques collaborant pour le contrôle et la commande d'entité physique. C'est une classe de systèmes complexes, de grande échelle, qui apparaissent dans des applications industrielles variées.

Un système cyber-physique est composé de sous-systèmes, dont la conception et le fonctionnement font intervenir différents corps de métiers.

Un pacemaker est un exemple type d'un système cyber-physique, c'est un dispositif électronique implanté dans le cœur d'un patient souffrant de bradycardie; il se caractérise par un rythme cardiaque trop bas par rapport à la normale, ou à l'inverse de tachycardie, c'est à dire d'un rythme cardiaque trop élevé, en l'absence d'effort physique significative. Le but d'un pacemaker est d'émettre des signaux électriques directement sur le cœur. Le cœur étant un muscle, la réception de ces signaux correctement dimensionnées vont corriger le mouvement.

La modélisation des systèmes cyber-physique est une tâche ardue liée directement à la complexité de ses systèmes qui sont constitués de plusieurs sous-systèmes et de différents composants interconnectés qui réalisent différentes fonctions.

Ce projet s'inscrit dans le cadre précédemment décrit, il consiste à se confronter à un problème de modélisation difficile, qui consiste à modéliser un système à partir d'une spécification réelle. Le système que nous proposons de modéliser est un pacemaker.

Ce mémoire est organisé en quatre chapitres. Le premier chapitre sera consacré à la présentation et la modélisation du système cyber-physique, ainsi que les domaines d'utilisation de chacun avec les caractéristiques et quelques exemples pour mieux comprendre.

Dans le deuxième chapitre, nous l'avons consacré pour la conception, nous commencerons à ce niveau d'abord par une petite introduction. Ensuite, nous présenterons l'ingénierie dirigée par les modèle où nous donnerons une vision sur les notions de modèle, méta modèle et méta méta-modèle avec une comparaison entre eux, après nous passerons à l'architecture dirigé par les modèles où nous parlerons du MDA en détail nous compterons ces principe, ces différents modèles et nous terminons par la transformation de ces modèles.

Le troisième grand titre sera consacré à la transformation des modèles où nous présenterons deux types de transformations « endogènes et exogènes », nous préciserons aussi comment faire ~~le~~ la transformation de type modèle vers code ainsi que de type modèle vers modèle, et

puisque cette transformation sera faite à l'aide d'outils et de langages ~~a~~ nous passerons en vue ses outils et ses langages de transformation.

Dans Le troisième chapitre, nous nous intéressons au processus de dérivation avec ces définitions et ces principes.

Pour le dernier chapitre nous l'avons consacré pour la réalisation, c'est-à-dire, la mise en œuvre du méta modèle avec présentation des outils utilisés, à savoir, AToM3 et Dia.

Ce chapitre a pour objectif de représenter les systèmes cyber-physiques (CPS), avec leur définition et leurs caractéristiques aussi la modélisation .

1. Les systèmes cyber-physiques

1.1 -Définition :

Un système cyber-physique en anglais « cyber-physical system », en bref on l'appelle CPS : est un système ou où des éléments informatiques collaborant pour le contrôle et la commande d'entité physique [1].Une génération de précurseurs des systèmes cyber-physiques est très importante donc elle peut être trouvée dans plusieurs domaines comme l'automobile, aéronautique, les processus chimique, l'énergie, la santé, l'infrastructure civile, la fabrication, le transport, les divertissements et les appareils électroménagères ce qui peut être synthétisé par la (Figure 1.1).

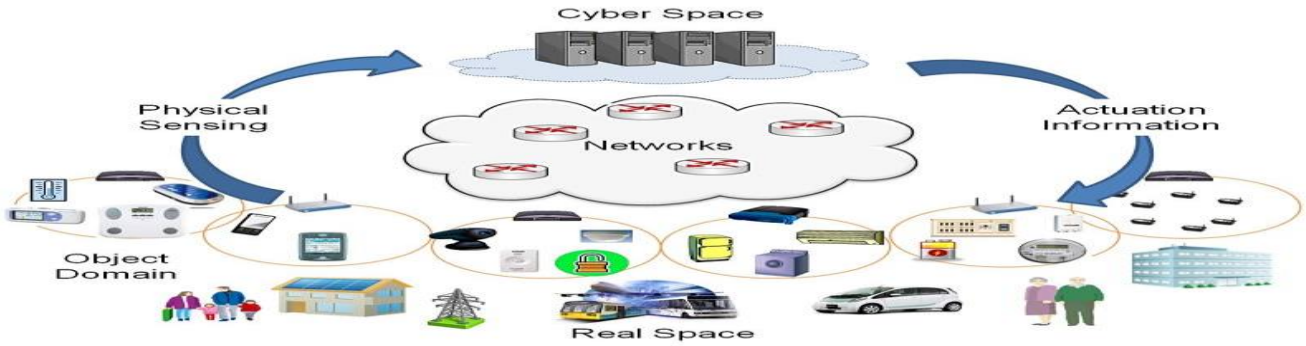

FIGURE 1.1 -domaine d'utilisation du CPS [1].

Cette génération est comme systèmes embarqués. Dans se derniers l'accent tend à être plus, sur les éléments informatiques et moins sur les liens entre les éléments informatiques et physiques.

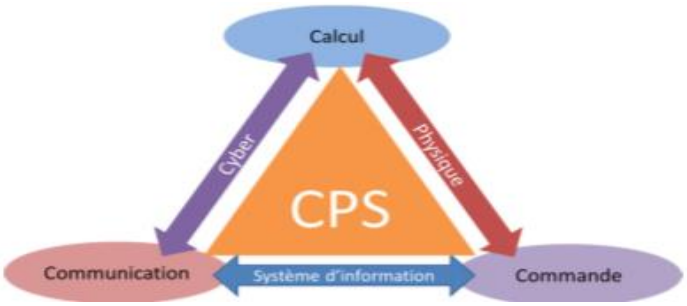

FIGURE 1.2 - Composantes d'un système cyber physique[1].

La déférence entre les systèmes cyber physiques et les systèmes embarqués traditionnelle c'est que un cps est généralement conçu comme un réseau d'éléments informatiques en interaction avec des entrées et des sorties physiques au lieu de dispositifs autonomes en interaction [2]. Cette notion est étroitement liée aux concepts de la robotique et des réseaux de capteurs.

Le développement dans les sciences et l'ingénierie permettront d'améliorer le lien entre les éléments de calcul et physiques, aussi faite une grande augmentation dans la capacité d'adaptation, l'autonomie, la fonctionnalité, l'efficacité, la fiabilité, la sécurité et finalement utiliser les systèmes cyber-physiques facilement. Ceci peut améliorer les capacités des systèmes cyber-physiques dans différentes dimensions on note [3] :

-**La réactivité** : par exemple pour l'évitement des collisions.

-**La précision** : par exemple de la chirurgie robotique

-**Travaille dans des environnements dangereux ou inaccessibles** : par exemple pour le sauvetage, la lutte contre les incendies.

-**La coordination** : par exemple pour le contrôle de la circulation aérienne dans le domaine militaire.

- **L'efficacité** : par exemple pour les bâtiments à énergie positive.

- **L'augmentation des capacités humaines** : par exemple pour les dispositifs de contrôle de santé personnels.

Les systèmes cyber-physiques sont un des deux piliers de l'Industrie 4.0 [5]. Pour cela la National Science Fondation (NSF) les a identifiés comme domaines de recherche prioritaires et très importants [6]. Depuis la fin de 2006, la NSF et d'autres agences gouvernementales ont financé des ateliers sur les systèmes cyber-physiques[7]Le 1er juillet 2019 l'Union européenne a lancé le projet CPS4EU[16] qui contient les grandes entreprises (VALEO, THALES, TRUMPF, RTE, LEONARDO et SCHNEIDER ELECTRIC).

1.1.1. Les caractéristiques les plus importantes [8] :

Les systèmes cyber-physique ont plusieurs caractéristiques comme Électronique, informatique, métrologie, automate, data fusion et mécanique en peut résumer tout ca dans la figure suivante :

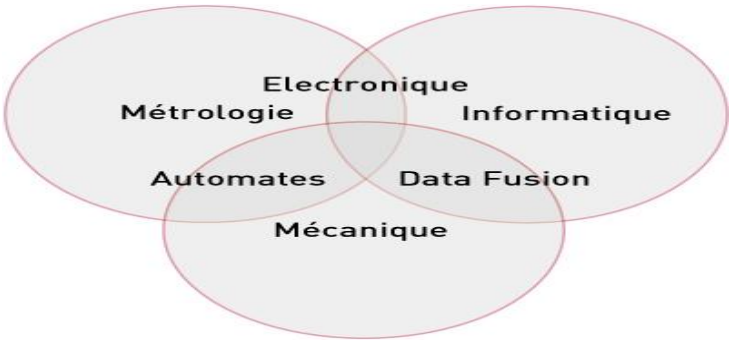

FIGURE 1.3 -Défférentes caractiristiques d'un CPS[8].

Chacune de ces caractéristiques est utilisée dans des cas différents :

Le domaine Électronique :

On cite :

-L'intelligence interne

- Circuits de protection (entre autres protection CEM)

- Utilisation sous conditions difficiles.

Le domaine Métrologie :

On cite :

-Haute précision de signaux

- Réduction des incertitudes de mesure.

Le domaine Informatique :

On cite :

-Connexion entre le monde physique et le monde numérique

- Interconnexion entre les systèmes et à l'extérieur

- Intégration dans le réseau de l'entreprise et aussi technologie de l'internet.

Le domaine Automates :

On cite :

- Connexion aux logiciels d'automates les plus courants

-Transfert seule des données appréciables et Transfert de données physique

Fusion de données :

On cite :

- La Combinaison et le calcul de données de différentes sources
- La Fusion de données multi-sensorielles.

Le domaine Mécanique :

On cite:

-Boîtier robuste contre les perturbations

-Résistance contre les chocs et l'intégration dans la machine

1.1.2 Quelques exemples de développements de CPS [9] :

Lorsque nous parlons de CPS nous devons parler tout d'abord de ses importants et utilisation dans les différents domaines (domaine d'automobile, aéronautique, l'énergie, la santé …ext) (parti précédente) . Donc comme une conséquence CPS ont fait l'objet de nombreux travaux de recherche lors de la dernière décennie, amenant assez régulièrement jusqu'au niveau du développement de prototypes fonctionnels. Dans ce qui suit nous essayons d'énumérer quelques exemples de développement.

Exemple 1:

Au début, les premières applications des CPS se sont basées sur les appareils de type « smartphone » pour déployer les applications. De ce fait, les applications d'assistance à la personne se sont développées, et particulièrement celles basées sur l'aide médicale. La vision de la « santé connectée » a pris de l'étalement ces dernières années, étalement grâce à l'essor de technologies connexes telles que les réseaux sans fil ou les capteurs. Ceci a permis le développement d'Equipements de Santé Personnels (Personal Health Devices (PHD)) qui visent à récupérer puis partager de l'information sur un réseau local ou Internet, tel que celui présenté sur la Figure 1.4 (Santos et al., 2015).

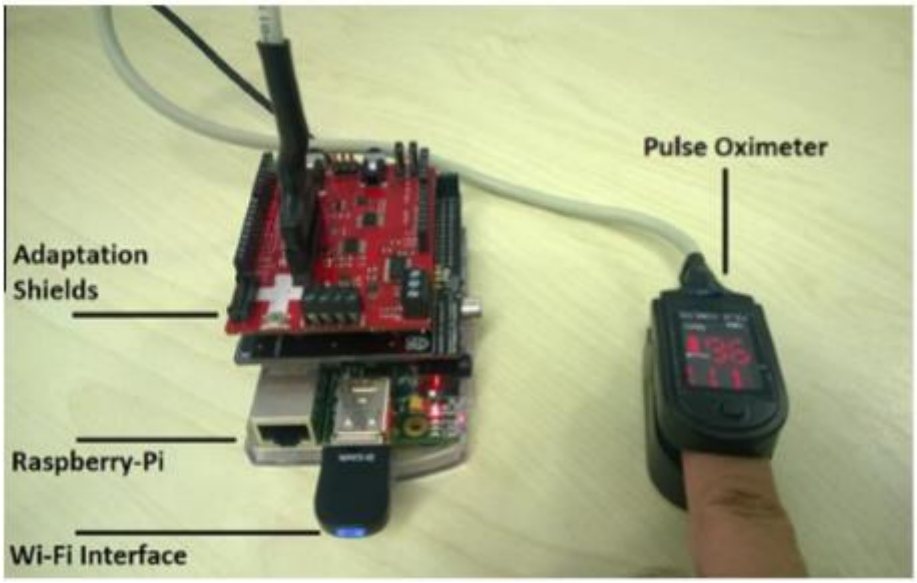

FIGURE 1.4-Prototype d'équipement personnel de santé (Santos et al., 2015)[9].

Exemple 2 :

Les CPS ont également de nombreuses applications dans le domaine de l'énergie (génération, distribution, consommation). Ces systèmes ont pour intérêt par exemple de diminuer les conséquences d'une défaillance.. Une ligne de 430km de long relie une turbine à vapeur d'eau à un bus infini. Le courant est mesuré de chaque côté de la ligne et est envoyé à un contrôleur central par le réseau. Ce contrôleur, par comparaison du courant sur les phases, détecte les éventuelles défaillances et agit sur les coupe-circuits, également via le réseau, ce qui constitue un CPS : {C2 ; Energy ; Full ; Ethernet}

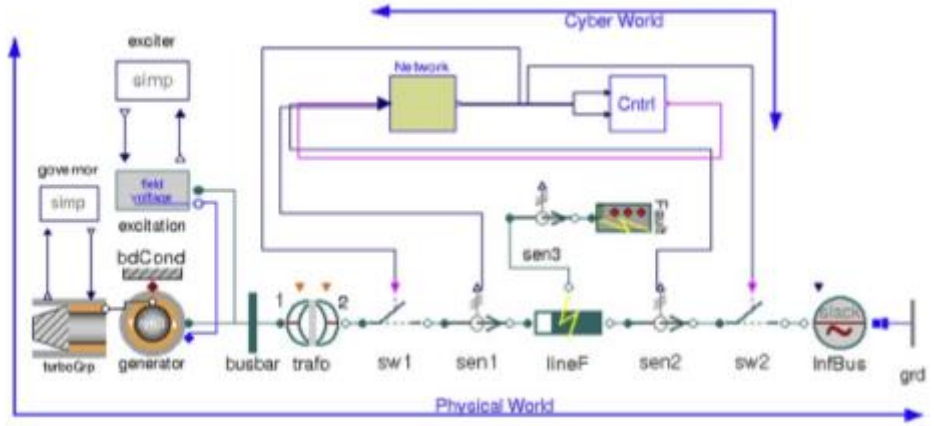

FIGURE 1.5 -Schéma de principe d'un système de transmission de puissance électrique (Al-Hammouri, 2012)[9]

1.2 La modélisation des systèmes cyber-physiques

1.2.1 Introduction :

Les systèmes cyber-physique ou embarqués sont des systèmes intégrant du logiciel et matériel, inclus dans des objets ou des systèmes qui ne sont pas perçus comme des ordinateurs par leurs utilisateurs.

La complexité croissante des systèmes informatiques entraîne des difficultés d'ingénierie pour les systèmes à base de composants, en particulier liées à l'optimisation, la validation et à l'analyse des performances et des exigences concernant la sûreté de fonctionnement. Des approches d'ingénierie guidée par des modèles sont de plus en plus utilisées dans l'industrie dans l'objectif de maîtriser cette complexité au niveau de la conception. Ces approches encouragent la réutilisation et l'automatisation du cycle de développement. Elles doivent être accompagnées de langages et outils capables d'assurer la conformité du système implémenté aux spécifications.

1.2.2 Architecture Dirigée par les Modèles [10] :

L'Architecture Dirigée par les Modèles (Le Model Driven Architecture, MDA)[11]est une démarche de développement proposée par l'Object Managment Group (OMG) [12]. Elle permet de séparer les spécifications fonctionnelles d'un système des spécifications de son implémentation sur une plate-forme donnée. A cette fin, le MDA définit une architecture de spécifications structurée en modèles indépendants des plates-formes (Platform Independent Model, PIM) et en modèles spécifiques (Plateform Specific Model, PSM).

L'approche MDA permet de réaliser le même modèle sur plusieurs plates formes grâce à des projections standardisées. Elle permet aux applications d'interopérer en reliant leurs modèles et supporte l'évolution des plates formes et des techniques. La mise en œuvre du MDA est entièrement basée sur les modèles et leurs transformations.

Cette approche consiste à manipuler différents modèles de l'application à produire, depuis une description très abstraite jusqu'à une représentation correspondant à l'implantation effective du système. Le processus MDA se décompose en trois étapes principales, représentées sur la figure (1.6) :

1. la définition d'un modèle de très haut niveau, indépendant des contraintes d'implantation : le PIM peut être raffiné afin d'ajouter différentes informations non fonctionnelles telles que la

gestion de la sécurité ; ces informations demeurent indépendantes de la plateforme d'exécution qui sera effectivement utilisée ;
2. le PIM est ensuite transformé pour prendre en compte les spécifications propres à la plateforme d'exécution ; le nouveau modèle est appelé PSM; le PSM peut être également raffiné afin de prendre en compte différents paramètres liés à l'environnement d'exécution.
3. Le PSM est ensuite utilisé pour générer une application exécutable basée sur les spécifications à partir desquelles le PIM a été construit

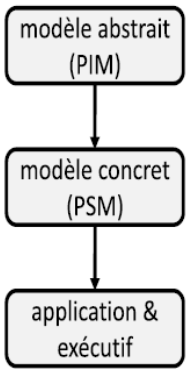

FIGURE 1.6 - L'approche MDA [12].

MDA propose une démarche intégrée permettant de rassembler tous les éléments pour la description d'une application. De cette façon, MDA vise la mise en place de différentes étapes de raffinement depuis la conception de l'application jusqu'à la production de code exécutable.

1.2.3 Modèles formels des applications :

L'approche MDA peut être instanciée en utilisant différents langages. C'est pourquoi un certain nombre de langages permettant cette description sont apparus et offrent à ce jour un certain nombre de fonctionnalités. La plupart des approches guidées par des modèles se basent soit sur :

• Le langage de modélisation unifié (Unified Modeling Language, UML), qui est un langage de modélisation à usage général et ses profiles comme MARTE (Modeling and Analysis of Real-Time and Embedded) . Si UML permet une grande expressivité, il ne transporte par en lui-même une sémantique précise pour décrire les architectures.

• Des langages de description d'architecture (Architecture Description Language, ADL), qui sont des langages propres à des domaines particuliers. Un ADL est un langage qui permet la modélisation d'une architecture conceptuelle d'un système logiciel et/ou matériel. Il fourni une syntaxe concrète et une structure générique (framework) conceptuelle pour caractériser

les architectures. Parmi les langages ADL on trouve le langage AADL qu'on va détailler dans la suite de ce chapitre.

• Des langages spécifiques, conçus pour répondre à des besoins particuliers de modélisation et d'analyse, par exemple Behavior Interaction Priority (BIP), Fractal et Ptolemy.

1.2.4 Le langage AADL [13] :

Parmi les ADL, le langage d'analyse et de description d'architectures (*Architecture Analysis and Design Language*, AADL), a fait l'objet d'un intérêt croissant dans l'industrie des systèmes embarqués critiques(comme Honeywell, Rockwell Collins, l'Agence Spatiale Européenne, Astrium, Airbus). AADL a été standardisé par la SAE (*International Society of Automotive Engineers*) en 2004, pour faciliter la conception et l'analyse de systèmes complexes, critiques, temps réel dans des domaines comme l'avionique, l'automobile et le spatial. AADL fournit une notation textuelle et graphique standardisée pour décrire des architectures matérielles et logicielles. Le succès d'AADL dans l'industrie est justifié par son support avancéà la fois pour la modélisation d'architectures reconfigurables et pour la conduite d'analyses. En particulier, le langage a été conçu pour être extensibleafin de permettre des analyses qui ne sont pas réalisables avec le langage de base.

AADL (Architecture Analysis & Design Language) est un langage de description d'architectures. Ce langage permet la conception et la description d'architectures temps-réel embarquées, en considérant simultanément tous les aspects logiciels et matériels sous-jascents. C'est la SAE (Society of Automotive Engineers) qui a préparé le standard international d'AADL depuis 2001, dont la version 1.0 est parue en 2004.

1.2.4.1 Principes du langage AADL :

AADL peut être exprimé selon différentes syntaxes. Le standard en définit trois : en texte, en XML, ainsi qu'une représentation graphique. Par cette multiplicité des syntaxes possibles, AADL peut être utilisé par de nombreux outils différents, graphiques ou non. Le développement de profils pour UML permet également d'envisager l'intégration d'AADL au sein d'outils de modélisation UML.

AADL a été créé avec le souci de faciliter l'interopérabilité des différents outils, c'est pourquoi la syntaxe de référence est textuelle. La représentation XML permet de faciliter la création de parseurs pour des applications existantes. La notation graphique se pose en complément de la notation textuelle, pour faciliter la description des architectures. Elle permet une représentation beaucoup plus claire que le texte ou XML, mais elle est moins expressive.

En tant que langage de description d'architecture, AADL permet de décrire des composants connectés entre eux pour former une architecture. Une description AADL consiste en un ensemble de déclarations de composants. Ces déclarations peuvent être instanciées pour former la modélisation d'une architecture.

1.2.4.2 Les composants AADL

Les composants AADL sont définis en deux parties : l'interface et les implantations. Un composant AADL possède une interface (component type) à laquelle correspondent zéro, une ou plusieurs implantations (component implémentation). Toutes les implantations d'un composant partagent la même interface.

a. Types et implantations

Le type d'un composant définit son interface tandis que l'implantation décrit les éléments (sous-clauses) de sa structure interne. Une implantation fait toujours référence à un type défini par ailleurs. AADL étant un langage déclaratif, l'ordre des déclarations n'importe pas : il est possible de déclarer une implantation avant le type correspondant. Voici un exemple illustratif.

```
processor myProcessor
end myProcessor;

processor implementation myProcessor.Intel
end myProcessor.Intel;

processor newProcessor
end newProcessor;

processor implementation newProcessor.newIntel
                         extends myProcessor.Intel
end newProcessor.newIntel;
```

FIGURE 1.7 -Types et implantations de composants AADL[13]

Il est possible d'étendre une déclaration de composant (type ou implantation) afin d'y ajouter ou d'en préciser des éléments. La figure 1.8 résume les mécanismes d'extensions possibles. Nous pouvons remarquer que l'implantation d'un type qui en étend un autre peut étendre une implantation du type initial.

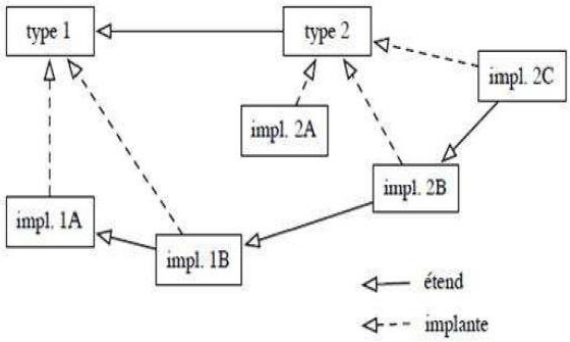

FIGURE 1.8 -Possibilités d'extension des composants [13]

L'extension de composant AADL ne correspond pas exactement à la notion d'héritage des langages objet : un composant et son extension constituent des composants distincts; il n'est pas possible d'utiliser l'un pour l'autre au sein d'une description architecturale. En revanche, les implantations d'un type de composant peuvent être substituées les unes aux autres dans la mesure où elles partagent la même interface aux autres composants.

b. Catégories de composants:

AADL définit plusieurs catégories de composants, réparties en trois grandes familles :

– **les composants logiciels** définissent les éléments applicatifs de l'architecture.

– **les composants de la plate-forme** d'exécution modélisent les éléments matériels.

– **les systèmes** permettent de regrouper différents composants en entités logiques pour structurer l'architecture.

1.Composants logiciels :

Il existe quatre catégories de composants logiciels :

– les données.

– les sous-programmes.

– les fils d'exécution (threads).

– les processus.

(a) : Données (b) : Sous-programmes (c) : Threads (d) : Processus

FIGURE 1.9 -Syntaxe graphique des composants logiciels [13].

2. Composants de plate-forme :

Il existe quatre catégories de composants de plate-forme :

– les processeurs.

– les mémoires.

– les bus.

– les devices (dispositifs).

(a) : Processeurs (b) : mémoires (c) : bus (d) : devices

FIGURE 1.10 -Syntaxe graphique des composants matériels[13].

1.3 Conclusion

Dans ce chapitre nous avons présenté système cyber physique avec ces caractéristique et Domain d'utilisation et en termine par la modélisation .

2.1 Introduction

Aujourd'hui, car l'augmentation de la taille et la complexité des logiciels développés de plus en plus rapidement on trouve des nouvelles techniques du génie logiciel qui sont apparues.

Dans ce contexte, Les grandes tendances actuelles sont basées sur l'utilisation de modèles pour permettre une montée en abstraction par rapport aux langages de programmation, et pour faciliter et simplifier notre travail. C'est Ingénierie Dirigée par les Modèles (IDM).

Dans ce chapitre, nous présentons une définition pour IDM et les concepts de base de la démarche IDM (modèle, Méta-Modèle, Méta-méta-modèle) en expliquant avec plus de détail l'approche MDA, ses principes, différents modèles et en termine par la transformation de ces modèles Ensuite, nous illustrons quelques notions fondamentaux autour de la transformation de modèles.

2.2 Ingénierie Dirigée par les Modèles

L'ingénierie dirigée par les modèles (IDM) ' Model Driven Engineering ' (MDE) en anglais, IDM c'est le résultat de l'évolution du processus de développement logiciel d'une utilisation "contemplative" à une utilisation "productive" des modèles, en outre c'est une approche parmi les approches qui présente la mécanisation du processus que les ingénieurs expérimentés suivent manuellement ,elle est plus globale et générale que l'architecture dirigée par les modèles , IDM est basée sur la transformation de modèle car ce dernier joue un rôle fondamental et considérée l'une des techniques prometteuses dans cette approche. les modèles étaient utilisés comme des éléments de conception, de discussion ou de documentation, l'idée de l'IDM est de les exploiter comme entrée du processus de développement.

Dans le coté pratique, cela nécessite de formaliser les modèles pour les rendre exploitables par la machine et de réaliser des programmes permettant de traiter des modèles.

Dans la terminologie de l'IDM, ces programmes sont regroupés sous le terme de transformations de modèles [14].

Dans 'IDM', les modèles sont utilisés de manière systématique tout le long du processus de développement d'un logiciel et ils doivent être suffisamment précis afin de pouvoir être transformés. Le processus de développement d'un logiciel est caractérisé par une séquence partiellement ordonnée de transformations de modèles. Chaque transformation prend des

modèles en entrée et produit des modèles en sortie, jusqu'à l'obtention d'artefacts exécutables [15].

2.2.1 Concepts fondamentaux

Cette partie définit les concepts de base de l'ingénierie des modèles et les relations qui existent entre ces concepts.

2.2.1.1 Modèle (m1)

Un modèle est une simplification de la réalité, Il décrit les informations de M0. De plus est une abstraction d'un système construite dans un but donné ,il contient un ensemble restreint d'informations sur un système et il est construit dans un but précis en ne contenant que les informations pertinentes vis-à-vis de l'utilisation qui sera faite du modèle [15].Ces modèles permettent de mieux comprendre le système que l'on développe.

- ✓ **Niveau M0 :** Le niveau M0 (ou **instance**) correspond au monde réel (réalité) . Ce niveau contient les informations réelles de l'utilisateur, instances du modèle du niveau M1.

2.2.1.2 Méta-Modèle (m2) : Langage de modélisation

Un méta-modèle (niveau M2) définit le langage de modélisation des modèles M1, il permet d'énoncer des modèles en définissant les concepts nécessaires et les relations entre eux. Donc, un modèle est une construction possible du méta-modèle dans lequel il est défini. Dans la littérature, un modèle est dit conforme à son méta-modèle.

2.2.1.3 Méta-méta-modèle (m3) : langage de méta-modélisation

Dans l'approche MDA, méta-méta-modèle (niveau M3) est composé d'une unique entité qui s'appelle le MOF. Ce dernier permet de décrire la structure des méta-modèles. Donc il est nécessaire d'avoir un méta-modèle pour interpréter un modèle, pour pouvoir interpréter un méta-modèle il faut disposer d'une description du langage dans lequel il est écrit : un méta-modèle pour les méta-modèles. C'est naturellement que l'on désigne ce méta-modèle particulier par le terme de méta-méta-modèle [14]. Pour limiter le nombre de niveaux d'abstraction, et éviter de définir un méta-méta-méta-modèle, les méta-méta-modèles sont généralement conçus pour être auto-descriptifs, c'est-à-dire capables de se décrire eux-mêmes.

Les figures 2.1 et 2.2 représentent les différentes relations qui existent entre les modèles, méta-modèles et méta-méta-modèles. On distingue sur ces figures les trois niveaux de modélisation M1, M2 et M3. Dans la littérature, et en particulier dans certains documents de l'OMG, le niveau M0 correspondant au système que représente un modèle du niveau M1. La nature de la relation qui existe entre le niveau M0 et M1 est donc différente de la nature des

relations existant entre les niveaux M1 et M2 et M2 et M3. Pour éviter ce problème d'homogénéité, le niveau M0 n'est généralement pas considéré.

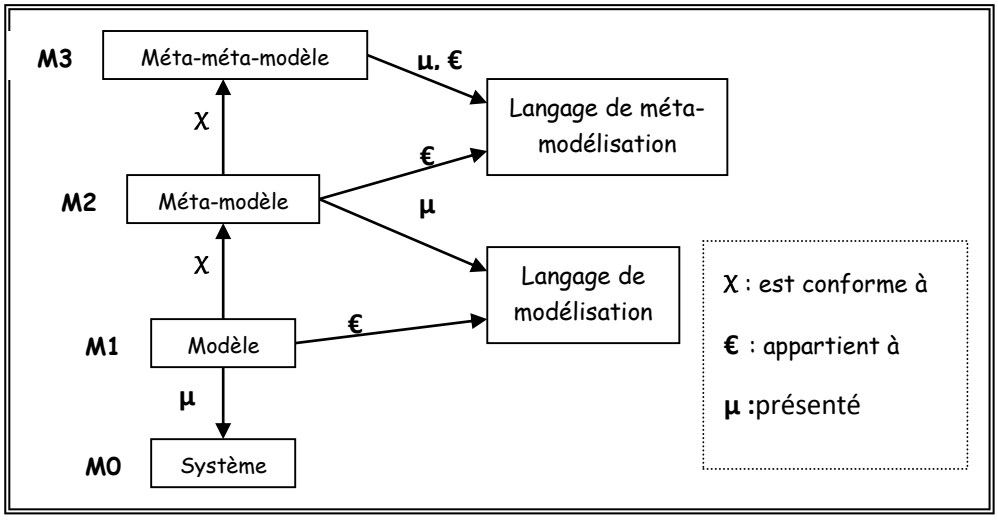

FIGURE 2.1 -Niveaux de modélisation[14].

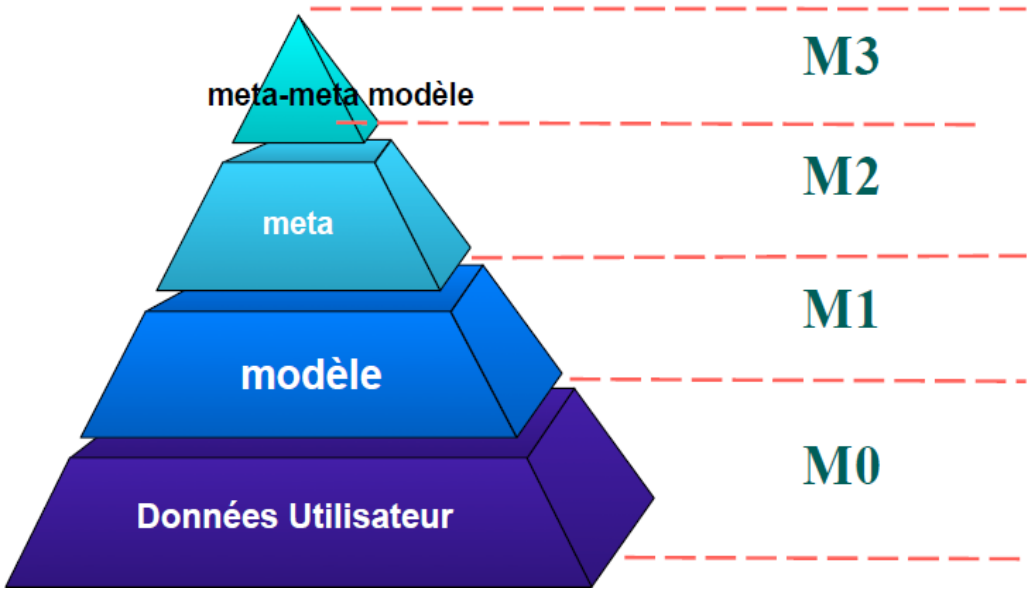

FIGURE 2.2 -autre image pour préciser les niveaux de modalisation [14].

On retrouve également dans ces figures, les trois relations de base de l'ingénierie des modèles: la conformité, la représentation et l'appartenance. Cela permet de mettre en évidence les relations qui existent entre modèles et langages.

2.2.2 Architecture Dirigée par les Modèles

MDA est une approche proposée et soutenue par l'Object Management Group (OMG) en novembre 2000.Cette approche peut être considérée comme une variante du MDE pour le génie logiciel. C'est un champ de développement, de maintenance et d'évolution de produits logiciels basé sur l'utilisation de modèles. Nous décrivons son principe et ses objets dans cette partie.

2.2.2.1 Principe du MDA

La démarche MDA a toutes les techniques de modélisation et de transformation qui réalise un changement notable dans la conception des applications. Elle est basée sur un principe qui sépare les spécifications fonctionnelles des spécifications de l'implantation sur une plate-forme donnée => interopérabilité des applications.

L'idée principale de MDA est d'aménager des modèles, d'abord d'analyse puis de conception, jusqu'au code, par transformations, dérivations et enrichissements successifs

L'OMG expose le langage déclaratif (à base de règles) "QVT" (Query/View/Transformation) pour exprimer les transformations de ces modèles.

Les principaux modèles sont :

- **CIM** modèle indépendant de calcul (computation independant model en anglais) : chargé de décrire les flux et les actions sur le système.
- **PIM** modèle indépendant des plates-formes (plateform independant model en anglais): chargé de décrire les traitements orientés métier.
- **PDM**modèle des plates-formes (plateform dependant model en anglais): chargé de décrire une architecture technique (plusieurs par projet)
- **PSM** modèle dépendant des plates-formes (plateforme specific model en anglais) : chargé de décrire les détails techniques liés à l'implantation pour une plate-forme

On peut les voire dans la figure 2.3 au-dessous.

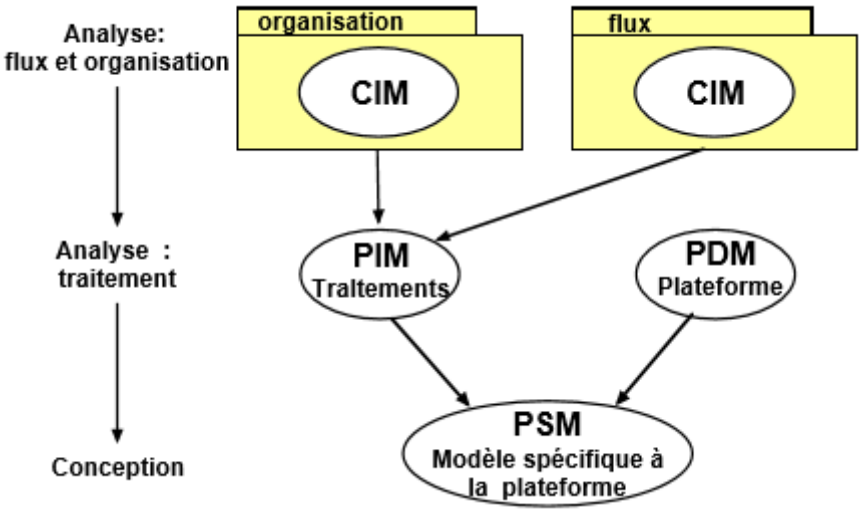

FIGURE 2.3 - Les principaux modèles[14].

2.2.2.2 Différents modèles du MDA

Comme nous avons déjà vue dans la section précédente l'Architecture Dirigée par les Modèles (Model Driven Architecture – MDA en anglais) : est une approche proposée par l'Object Management Group (OMG),il est composé de plusieurs modèles qui vont servir dans un premier temps à modéliser l'application, puis par transformations successives à générer du code ces modèles sont :CIM , PIM,PDM,PSM

Le CIM (Computation Independent Model) :

Le CIM est un modèle parmi les modèle MDA, en outre c'est le modèle d'analyse de base du métier ou du domaine d'application, la disposition du CIM consiste à décrire le produit indépendamment de tout système informatique mais en respectant les exigences des utilisateurs. Cette indépendance technique permet de capitaliser le savoir faire dans le CIM en faisant abstraction de la technologie d'implantation [16].Aussi il lui permet de garder tout son intérêt au cours du temps et il est modifié uniquement si les connaissances ou les besoins métier changent.

Les exigences modélisées dans le CIM seront prise en compte dans les constructions des PIM (Platform Independent Model) et des PSM (Platform Specific Model)

Le PIM (Platform Independent Model)

Le PIM supplément c'est un modèle de conception qui décrit le système indépendamment de toute plate-forme technique et de toute technologie utilisée pour déployer l'application, il est connexe avec CIM ,pour cela une fois le

CIM établi, il est possible de concevoir les modèles d'analyse et de conception (PIM) en respectant les conditions d'utilisation du produit spécifiées dans le CIM.

Le PIM intègre les aspects technologiques et architecturaux sans manifester les détails de l'utilisation du produit sur sa plate-forme technique et caractérise le fonctionnement des entités et des services. Pour coté pratique, il existe différentes méthodes qui permettent de construire des modèles d'analyse et de conception telles que Merise, Booch et OMT.

Toutefois, le MDA recommande d'utiliser UML pour représenter les PIM, a ce niveau, le formalisme utilisé est un diagramme de classes qui peut-être couplé avec un langage de contrainte comme OCL (Object Constraint Language) [17].

Le PDM (Plateform Description Model)

Le pdm supplément c'est un modèle de transformation pour permettre la circulation du PIM vers le PSM**,** car il contient des informations pour la transformation des modèles vers une plateforme donc Apres la définition du PIM, l'architecte doit choisir une plate-forme technique qui permet d'implanter le produit avec la qualité architecturale décrite dans le PIM. Cette plateforme doit être décrite dans le modèle de pdm afin de permettre son intégration dans la construction du produit. . L'OMG ne donne pas plus de précisions sur le PDM dans la description du MDA. Dans la littérature, on pense que chaque fournisseur de plate-forme devrait proposer le PDM [16].

Une plate forme est un ensemble de sous-système et de technologies qui fournissent un ensemble cohérent de fonctionnalités à travers des interfaces qui peuvent être employés par n'importe quelle application soutenue par cette plateforme sans souci les détails de la façon dont la fonctionnalité fournie par la plateforme est mise en application [18].

Le PSM (Platform Specifique Model)

Une fois le PIM et le PDM connus, on peut construire le modèle du code ou modèle de conception effective du produit : le PSM. Le PSM dépend de la plate-forme technique choisie par l'architecte, il indique comment le produit sera utilisé sur cette plateforme et il sert essentiellement de base à la génération du code exécutable vers la plate-forme choisie [19].

2.2.2.3 Transformation de modèles MDA

Pour préparer et faciliter la génération de code vers la plate-forme technique choisie, MDA consiste à s'écouler progressivement des PIM aux PSM. Cette circulation des PIM aux PSM est une transformation de modèles. Le MDA identifie différents types de transformations de modèles dans le cycle de vie et de développement d'un produit. La figure 2.4 ci-dessous synthétise les différentes transformations possibles des modèles.

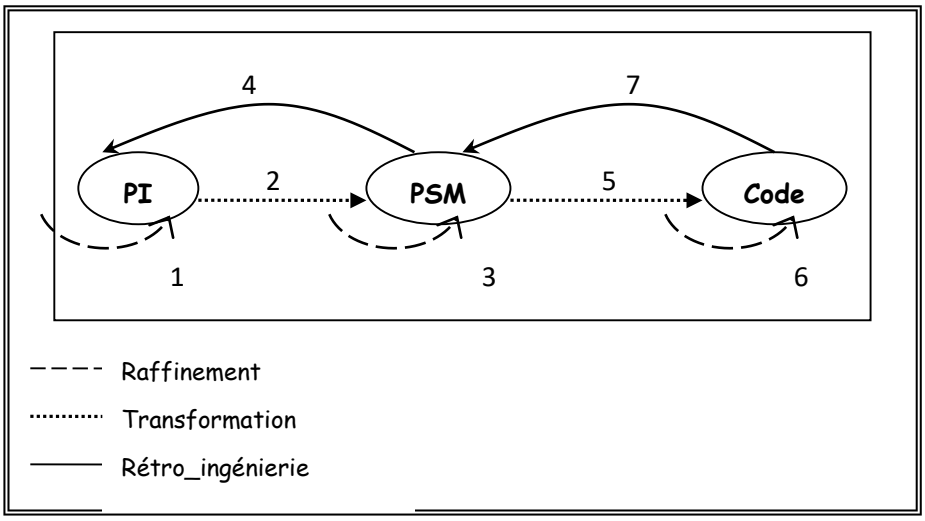

FIGURE 2.4 - Transformation des modèles MDA[19].

Le niveau de modèle le plus abstrait est représenté par Le PIM .Par passages consécutifs, les modèles deviennent de plus en plus concrets jusqu'à l'obtention du code. Ce code peut être assimilé à un PSM exécutable, de même que la génération de code n'est pas toujours considérée comme une transformation de modèle. La figure 2.4 montre aussi qu'il est possible à partir du code de recréer un PSM puis un PIM [17].

2.3 Transformation de modèles

L'ingénierie des modèles a deux principaux artefacts sont les modèles et les transformations de modèles. Généralement, on appelle *transformation de modèles* tout programme dont les entrées et les sorties sont des modèles. Maintenant plusieurs axes de recherche pour la transformation des modèles sont explorés. Cette section propose un rapide tour d'horizon des principes de base de transformation de modèles.

2.3.1 Définitions générales

Dans l'IDM, un ensemble de modèles cibles est génère par une transformation de modèles s'appuyer sur d'un ensemble de modèles sources conformément à une définition de

transformation. Les modèles sources et cibles sont décrites dans des méta-modèles précis. Ces méta-modèles peuvent être identiques ou différents. Les modèles sources et cibles peuvent appartenir à un même niveau d'abstraction ou bien différents [22].

2.3.1.1 Transformations endogènes et exogènes

On compte deux types de transformations de modèles par rapport à la différenciation des méta-modèles de définition des modèles sources et cibles, Transformations endogènes et exogènes :

1- Une transformation endogène : si les modèles sources et cibles sont définis dans le même méta-modèle.

2- Une transformation exogène : si les modèles sources et cibles sont définis dans déférent méta-modèle

2.3.1.2 Transformations horizontales et verticales

On compte deux types de transformations par rapport à la différenciation du niveau d'abstraction des modèles sources et cibles : les transformations horizontales et les transformations verticales. Une transformation est dite :

1- Horizontale : si les modèles sources et cibles appartiennent au même niveau d'abstraction. On cite comme un exemple typique de transformation horizontale: Le raffinement**.**

2- Verticale :si les modèles sources et cibles appartiennent à des niveaux d'abstractions différents. Par exemple la génération de code.

2.3.2 Concepts de base et principes

Une transformation est définie sur la base des méta-modèles de sa source et de sa cible. Les méta-modèles et les modèles (source, cible, y compris le modèle de la transformation) sont enregistrés dans un ensemble de référentiels. Cette transformation est exécutée à l'aide d'un moteur de transformation. On peut voir ca dans La figure 2.5 qui résume les concepts de base et le principe d'une transformation de modèles dans les différents outils.

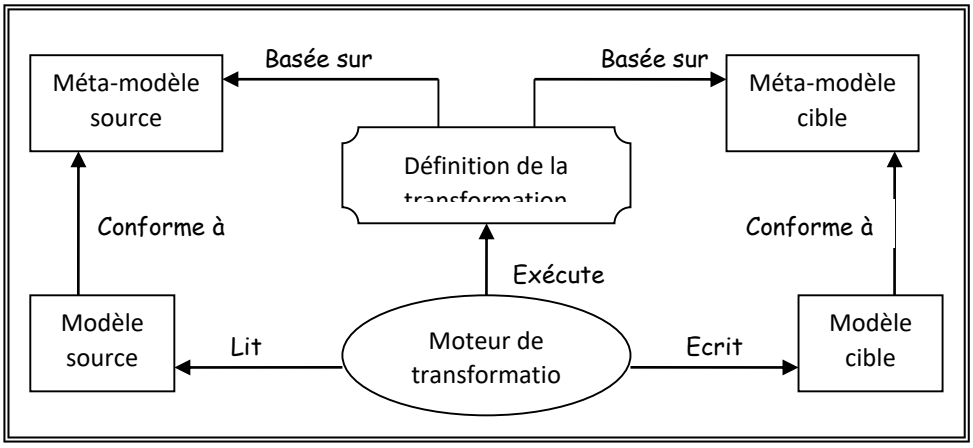

FIGURE 2.5 - Concepts de base et principe des transformations de modèles[19].

On rappelé que un moteur de transformation est un outil parmi les outils qui est susceptible de lire un modèle dans un référentiel donné et permet d'appliquer la transformation pour générer une solution conforme à la définition de la transformation. cette solution générée est soit un modèle, soit du code (texte). Pratiquement, on parle de la transformation de type « modèle vers modèle », dans le premier cas, et de la transformation de « modèle vers code » dans le deuxième cas [16].

2.3.2.1 Transformation de type modèle vers code :

Cette Transformation est un cas particulier de transformation de « modèle vers modèle » dans laquelle le méta-modèle de la cible correspond à la grammaire du langage de programmation cible ce type de transformation est utilisé pour générer du code dans un langage de programmation particulier (Java, C++ …etc) à partir d'un modèle source [16].
Puis en passe au deuxième type de transformation.

2.3.2.2 Transformation de type modèle vers modèle :

ce type de Transformation est basée sur les modèles dans les deux cotés ,il est utilisé pour réaliser une activité de génération automatique de modèles. Par exemple la dérivation de lignes de produits, qui est l'objectif du travail présent, le raffinement de modèles, la création d'un PSM à partir d'un PIM dans le MDA,…etc.

2.3.3 Caractéristiques d'une transformation de modèles

Une transformation de modèles est caractérisée par plusieurs éléments, tel que :Les règles de transformation , La portée des règles , La traçabilité , La directivité:

- ✓ **Les règles de transformation :** Une règle de transformation renferme deux parties : une partie gauche (LHS : Left-Hand Side) et une partie droite (RHS : Right-Hand Side).
  partie gauche : décrit les accès aux éléments du modèle source .
  partie droite : exprime les expansions (création, modification, suppression d'éléments) dans le modèle cible.
  Chaque partie peut être décrite en combinant des variables, des patterns et des expressions logiques décrivant des calculs et des contraintes sur les éléments de modèles.

- ✓ **La portée des règles :** Permet de limiter les parties d'un modèle (source ou cible) qui peuvent participer à la transformation.

- ✓ **La traçabilité :** Autorise de garder la trace des liens entre éléments source et cible, d'analyser l'influence d'un changement d'un modèle sur un autre et de synchroniser entre les modèles.

- ✓ **La directivité:** Les transformations peuvent être unidirectionnelles ou bidirectionnelles.
  Les transformations unidirectionnelles : peuvent s'exécuter dans un seul sens : le modèle cible est généré (crée ou modifie) à partir du modèle source, mais il est impossible de reconstituer le modèle source à partir du modèle cible. Par contre Les transformations bidirectionnelles s'exécutent dans les deux sens : le modèle cible est généré à partir du modèle source et vis versa.

2.3.4 Outils et langages de transformation

L'IDM réalise une grande réussite dans le domaine de l'ingénierie des logiciels grâce aux transformations de modèles. Cette réussite permet l'apparition de nombreuses techniques et outils de transformations de modèles talque : AGG, AToM3, MTL. Nous intéressons dans ce mémoire la par l'outil AToM3 qui sera présenté dans le dernier chapitre.

Dans ce domaine la Jean-Marc Jézéquel propose une classification intéressante dans laquelle, il regroupe les différents outils de transformations de modèles dans cinq catégories présentées ci-après :

1- Les outils génériques :

Ce type de outils faire le regroupent principalement les techniques de transformation d'arbres et de graphes. Ils permettent de transformer des modèles qui prennent la forme de graphes dirigés, attribués, à nœuds et arcs étiquetés. Il existe plusieurs systèmes de transformation de graphes qui permettent de réaliser des transformations de modèles, par exemple : ATOM3 et AGG.

2-Les outils intégrés aux ateliers de génie logiciel :

Les outils intégrés aux ateliers de génie logiciel sont majoritairement des outils propriétaires qui sont commercialisés par les vendeurs des ateliers de génie logiciel. On cite par exemple, l'outil Objecteering de Objecteering Software. Le profite de ces outils est leur intégration dans ces ateliers qui sont matures. Cette intégration poussée à un inconvénient : ces outils de transformation de modèles sont la plupart du temps propriétaires, non standardisés. De plus, ils ne sont que des fonctionnalités additionnelles pour leurs ateliers de génie logiciel. Ils n'ont pas été définis spécifiquement pour l'application de développements dirigés par les modèles [15].

3-Les outils spécifiques à la transformation des modèles :

On trouve de nombreux outils universitaires et industriels dans cette catégorie. Le standard QVT de l'OMG est l'outil de référence de cette catégorie. Dans le milieu universitaire on trouve par exemple les outils ATL (implante le standard QVT) et MTL de l'INRIA,…etc. comme des points forts de cet outil, on cite :

- la simplicité de développement et de maintenance des transformations
- la puissance d'expression et la composition des règles de transformation [15].

4-Les outils de méta-modélisation supportant les transformations :

Pour construire des modèles de transformations orientés objets et MDE pour exécuter ces modèles de transformations ce type de outil utilisent l'approche de programmation par objets. Dans ces outils, une transformation de modèles est tout simplement un méta-programme exécutable. Les outils de méta-modélisation sont moins nombreux que les outils spécifiques aux transformations de modèles [15].

5-Les outils associés aux langages de programmation :

Finalement en termine avec un autre type de outil c'est les outils associés aux langages de programmation,ce dernier est disponible sous forme d'APIs (Application Programming

Interface ou interface de programmation) dans des langages de programmation habituels tels que les langages Visual Basic, C++, C#....ext . L'atout majeur de ces outils est relatif au fait que les programmeurs de ces langages n'ont pas besoin d'apprendre de nouveaux langages pour définir des transformations de modèles. Mais, ces outils ne sont pas vraiment adaptés pour méta-modélisation et l'élaboration de transformations complexes.

2.3.5 Classification des outils de transformation

Les travaux réalisés dans le domaine de la transformation de modèles ne sont pas récents et peuvent être chronologiquement classés selon plusieurs générations en fonction de la structure de donnée utilisée pour représenter le modèle

- **Génération 1 : Transformation de structures séquentielles d'enregistrement**

  Dans ce cas un script spécifie la manière de récréation d'un fichier d'entrée en un fichier de sortie. Bien que ces systèmes soient plus lisibles et maintenables que d'autres systèmes de transformation, ils nécessitent une analyse grammaticale du texte d'entrée et une adaptation du texte de sortie.

- **Génération 2 : Transformation d'arbres**

  Ces méthodes permettent de faire un parcours d'un arbre d'entrée au cours duquel sont générés les fragments de l'arbre de sortie.

- **Génération 3 : Transformation de graphes**

  Ces méthodes permettent des transformations du modèle, c'est a dire un modèle en entrée (graphe orienté étiqueté) est transformé en un modèle en sortie. Ces approches visent à considérer l'opération de transformation comme un autre modèle conforme à son propre méta-modèle (lui-même défini à l'aide d'un langage de méta-modélisation).

2.3.6 La validation

Les transformations de modèles est très considérable dans le modèle d'ingénierie dirigée par les modèles. Elles permettent l'automatisation de certaines activités de développement aussi et la réutilisation de savoir faire. Pour assurer la qualité du processus de développement, il est nécessaire d'assurer la qualité des transformations de modèles. En effet, il existe peu de techniques de validation spécifiques à ce domaine. Pourtant, le besoin de telles techniques est clairement identifié dans la littérature: une condition pour que l'ingénierie des modèles tienne ses promesses en terme de qualité est d'assurer la validation des modèles et des transformations utilisées [14].

- **Validation de modèles**

Dans le contexte du développement basé sur les modèles, il existe deux approches pour la validation des modèles. La première approche est la validation directe des modèles qui s'appuyer à vérifier ou à tester les modèles. Mais cette technique a un inconvénient est qu'elle impose de disposer d'un environnement permettant de simuler ou d'exécuter des modèles. Pour la deuxième approche tire avantage du fait que les modèles sont utilisés pour produire une partie de l'application développée. L'idée est de tester le code produit à partir des modèles plutôt que de tester directement les modèles. Cette technique a des avantages aussi des inconvénients pour avantage est qu'il n'est pas nécessaire de disposer d'un environnement modélisation exécutable. Son inconvénient est que la distance entre les modèles et le code produit est potentiellement importante et que le passage de l'un à l'autre peut introduire des erreurs [14].

- **Validation de transformations de modèles**

Dans le domaine de l'ingénierie des modèles, beaucoup travaux s'intéressent aux langages et techniques pour le développement de transformations de modèles mais parmi ces travaux très peu s'intéressent aux problèmes de la validation. Pratiquement, les transformations de modèles sont implantées comme des programmes et peuvent donc être validées comme n'importe quel autre programme. Cependant, les transformations de modèles ont la particularité de manipuler des modèles qui sont eux-mêmes décrits par des méta-modèles.

2.4 Conclusion :

Dans ce chapitre nous avons passé en vue l'ingénierie dirigée par les modèles ainsi que les techniques de transformation de modèles.

3.1 Introduction :

L'objectif de ce chapitre est de présenter l'étude de cas d'un système cyber-physique. Nous avons retenu comme cas d'étude un pacemaker dont les spécifications fournies par la société Boston Scientific [20]

3.2 Le cœur:[20]

Le rôle du cœur est de permettre au sang d'irriguer les différents organes du corps humain. Il agit comme une pompe fonctionnant sans discontinuer tout au long de la vie. Ainsi, on peut ramener le fonctionnement du cœur à celui d'un système mécanique et électrique.

Le système mécanique du cœur (la pompe) est stimulé électriquement pour déclencher le mouvement des muscles. Le cœur est formé de quatre compartiments : les ventricules et atria (ou oreillettes) gauche et droit. Ces compartiments se contractent et se retractent périodiquement sous le contrôle de stimuli électriques. Les atria forment une unité, les ventricules une seconde unité.

Chacun de ses mouvements entraîne une séquence d'événements collectivement appelés la "révolution cardiaque". Celle-ci consiste en trois étapes majeures : la systole auriculaire, la systole ventriculaire et la diastole :

– Au cours de la systole auriculaire, les oreillettes se contractent et éjectent du sang vers les ventricules (remplissage actif). Une fois le sang expulsé des oreillettes, les valves auriculo-ventriculaires entre les oreillettes et les ventricules se ferment. Le sang continue tout de même à affluer dans les oreillettes. Ceci évite un reflux du sang vers les oreillettes. La fermeture de ces valves produit le son familier du battement du cœur.

– La systole ventriculaire implique la contraction des ventricules, expulsant le sang vers le système circulatoire. En fait, dans un premier temps, très bref, les valvules sigmoïdes sont fermées. Dès que la pression à l'intérieur des ventricules dépasse la pression artérielle, les valvules sigmoïdes s'ouvrent. Une fois le sang expulsé, les deux valves sigmoïdes - la valve pulmonaire à droite et la valve aortique à gauche se ferment. Ainsi le sang ne reflue pas vers les ventricules. La fermeture des valvules sigmoïdes produit un deuxième bruit cardiaque plus aigu que le premier. La pression sanguine augmente.

– Enfin, la diastole est la relaxation de toutes les parties du cœur, permettant le remplissage (passif) des ventricules (plus de 80% du remplissage dans les conditions usuelles), par les oreillettes droite et gauche et depuis les veines cave et pulmonaire.

Les oreillettes se remplissent doucement et le sang s'écoule dans les ventricules.

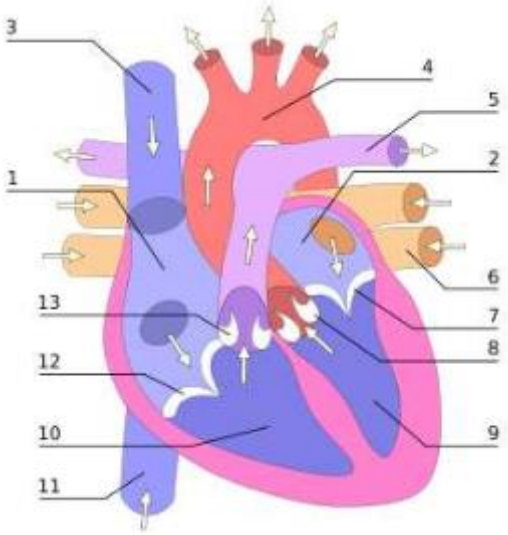

FIGURE 3.1 -Schéma du cœur[21].

En fonctionnement normal, le corps humain a en charge la stimulation du coeur via des mécanismes de contrôle par le système nerveux central et le système hormonal.

Ces deux systèmes jouent le rôle d'un pacemaker "naturel". Deux nœuds du système nerveux jouent un rôle clé dans la stimulation électrique: le nœud sinusal et le nœud atrio-ventriculaire. Le nœud sinusal reçoit une première décharge électrique, de l'ordre du milliVolt qui est ensuite transmise au nœud atrio-ventriculaire qui l'amplifie, stimulant les ventricules.

Un pacemaker peut être implanté à un patient dans le cas où le corps humain ne peut assurer seul un fonctionnement régulier du cœur, cohérent avec l'effort physique.

3.3 Pacemaker

Un pacemaker est un dispositif électronique implanté dans le cœur d'un patient souffrant de bradycardie (du grec bradus = lent et kardia = cœur) ; il se caractérise par un rythme cardiaque trop bas par rapport à la normale, en général inférieur à 60 pulsations par seconde, ou à l'inverse de tachycardie, c'est à dire d'un rythme cardiaque trop élevé, supérieur à une centaine de pulsation en l'absence d'effort physique significative.

Le but d'un pacemaker est d'émettre des signaux électriques directement sur le cœur. Le cœur étant un muscle, la réception de ces signaux correctement dimensionnées vont corriger le mouvement.

Les éléments de bases d'un pacemaker sont :

– **Sondes** : Un ou plusieurs fils métallique enroulé, le plus souvent deux qui transmettent un signal électrique entre le cœur et le pacemaker. Chaque sonde peut avoir un ou deux points de contacts avec le cœur, on parle alors de sonde unipolaire ou bipolaire.

– **Générateur de pulsation** : Le Pacemaker est à la fois la source de puissance, mais aussi le contrôleur de des battements du cœur, et son moniteur. Il contient donc un accumulateur et une unité générant un signal et analysant les battements du cœur, et éventuellement émettant un signal électrique. Il est implanté au plus près du cœur.

– **Moniteur-contrôleur externe** (Device Controller Monitor DCM) : Il s'agit d'une unité externe qui interagit avec le pacemaker via une connexion sans fil. Il contient la partie applicative du pacemaker, qui prend la décision d'émettre ou non un signal en fonction des battements du cœur du patient.

– **Accéléromètre** : L'accéléromètre, placé dans le pacemaker mesure l'accélération et les mouvements du corps du porteur, afin d'ajuster les stimulations.

Dans le cas d'un pacemaker à électrode simple, celle-ci est attaché à l'atrium pour le ventricule droit. Le pacemaker dispose de plusieurs modes opérationnels pour contrôler le mouvement du cœur, rappelées dans le document de spécification [Bos 07].

Ces paramètres sont liés à des contraintes temps réel, et de causalité (action/réaction) permettant de contrôler le rythme cardiaque.

Cette présentation rapide du pacemaker ne couvre pas ces nombreux cas d'utilisation, et les multiples interactions qui peuvent survenir entre patient, pacemaker et équipe médicale. En particulier, nous n'avons pas présenté les phases implantatoires, diagnostique et cas d'erreurs associés.

FIGURE 3.2 - Pacemaker de Boston Scientific [20]

3.4 Spécification du cas d'étude [20] [21]

Dans cette section, nous nous intéressons avant tout à la description du pacemaker du document de spécification [20]. Cette section vise aussi à introduire les exigences du système qui seront ensuite exploitées par les différents langages de modélisation proposés dans cet ouvrage. Les exigences sont numérotées, et indiquées comme suit :

(1)/S (**But du client**) Le pacemaker fonctionne correctement

Un identifiant dans la marge "(XX)/Y" indique que le paragraphe concerné est une exigence, définie en langage naturel . Un texte en gras en début de paragraphe définit la catégorie du besoin. Ainsi, l'exigence 1 définit une exigence naïve : le pacemaker fonctionne correctement. Le suffixe indique à quel niveau se place l'exigence, "S" pour le système, "P" pour le pacemaker lui-même.

Il nous faut maintenant raffiner cette exigence, et clarifier ce que "correctement" signifie. C'est l'objet des sections suivantes.

3.5 Définition du système

Le pacemaker auquel nous nous intéressons répond aux besoins suivants des patients :

– implantation

– phase ambulatoire, lors de la phase d'étalonnage du système

– suivi du patient,

– retrait (explantation) du pacemaker

Ce système fournit des stimulations dont le rythme s'adapte au patient, sur les deux chambres du cœur ; des données sur son fonctionnement dans un journal de suivi ; et un diagnostique de son bon fonctionnement.

Les fonctions d'analyse permettent de mesurer les éléments suivants : impédance des sondes, seuil de pulsation, mesure des ondes P et R3, état de la batterie, émission d'une pulsation, étalonnage de l'accéléromètre.

Le pacemaker est composé de trois composants principaux : le pacemaker a proprement parlé, c'est-à-dire le générateur de pulsations (PG – pusle generator) ; le

contrôleur-moniteur (DCM – Device Controller Monitor) et son logiciel de contrôle associé ; les sondes. L'aimant cardiaque en forme de beignet (un donut) est un composant mineur.

Le pacemaker (PG) contrôle et régule les pulsations cardiaques du patient, fournissant ainsi une thérapie pour les bradycardies. Le PG fournis un moyen programmable pour stimuler une

ou deux chambres cardiaques, adaptés au rythme du patient, que ce soit de manière permanente ou temporaire.

Dans le mode adaptable, un accéléromètre est utilisé pour mesurer l'activité physique du patient. Cette mesure est utilisée pour calculer la stimulation à émettre.

Le PG est programmé et interrogé par télémétrie par le contrôleur (DCM). Ceci permet au médecin de changer le mode opérationnel ou certains paramètres de manière non intrusive après implantation.

Le PG fournit les données suivants : données des capteurs ; histogramme des atriums et ventricules.

Le PG, en conjonction avec le DCM fournit les informations de diagnostique suivantes : télémétrie en temps réel ; électrogrammes (EGMs) ; mesures des ondes P et R; impédance des sondes ; état des batteries.

Le contrôleur-moniteur (DCM) est le support maître de la thérapie ; il est utilisé par les différentes équipes médicales (implantation, ambulatoire, suivi). Il communique par un protocole de communication et une couche matérielle dédiée. Le DCM est composé d'une plateforme matérielle et d'un bloc logiciel programmable.

Le DCM fournit les fonctionnalités suivantes : programmer et interroger le pacemaker; envoyer un ordre de stimulation immédiates ; gestion de l'historique des valeurs des capteurs ; affichage de l'historique et des informations d'étalonnage.

Par ailleurs, il dispose de fonctions auxiliaires : indicateur de bon fonctionnement (LED), acquisition de données pour un électrocardiogramme; suivi de l'état de la batterie ; sortie vers un dispositif d'affichage, etc. Par la suite, nous ne considérons pas ces fonctions.

Les sondes implantées sur le cœur du patient permettre de détecter l'activité électrique du cœur, et le stimulent. Les sondes sont connectées au PG via une connectique adaptée.

Toutes les sondes bipolaires IS-1 sont supportées.

Le pacemaker et les sondes sont implantés par une équipe médicale à l'hôpital. Le suivi des patients est effectué par des infirmiers et techniciens, sous la supervision d'un médecin.

Le pacemaker est conforme aux exigences en matière de sûreté électrique. Il doit fonctionner lors des différentes étapes de son cycle de vie, qui peuvent être source d'interférences : fluoroscope, machines de contrôle lors de l'anesthésie, poches d'eau, défibrillateurs, etc.

3.6 Cycle de vie du système

(2)/S Le cycle de vie du pacemaker est le suivant

– **Phase pré-implantatoire** Le système est assemblée suivant les bonnes pratiques édictées par les organismes de certification et de régulation sanitaires. Durant cette phase, les paramètres nominaux du PG sont configurés.

– **Implantation** Le pacemaker est placé à l'intérieur du patient. Lors de cette phase, l'unité DCM est utilisée pour interrogé le système, vérifier les batteries, tester le pacemaker, le configurer, programmer le système, évaluer les différents paramètres lues sur les sondes.
La procédure implantatoire prévoit les étapes suivantes : vérification de tous les équipements, implantation des sondes, évaluation des signaux au niveau des sondes, programmation du système avant implantation, mise en place des sondes dans le corps, connexion des sondes et test, implantation du PG.

– **Suivi préliminaire** durant cette phase, une série de tests peut être effectuée : interroger le système pour connaître ses paramètres, le reprogrammer, imprimer le rapport d'état du système pour archivage auprès du dossier patient.

– **Suivi de routine** le système programmable peut effectuer les actions suivantes, dites de routine : interroger l'état du système, vérifier l'état de la batterie, des paramètres de bradycardie du patient, mesurer les ondes P et R, tester le seuil de stimulation et d'impédance des sondes, passer en revue de l'historique des capteurs et histogrammes associés et les effacer, imprimer le rapport de tests. Si les paramètres du PG sont modifiés, ceux-ci peuvent être vérifiés avant leur implantation.

– **Ambulatoire** les fonctions de stimulations et de mesure doivent être disponibles.

– **Explantation** une fois le PG retiré, il est nettoyé et retourné à son fabriquant.
Une analyse de panne est éventuellement pratiquée. L'état du PG, lorsqu'il est retiré, dépend des pannes qu'il a pu suber et/ou de l'état de la batterie.

– **Destruction** le pacemaker est renvoyé à son fabriquant. Il ne doit pas être incinéré du fait de certains composants pouvant exploser en présence de fortes chaleurs.

3.7 Exigences du système

Dans la suite, nous considérons le système pacemaker pris seul, dans le cas du traitement de la bradycardie.

(3)/S Le pacemaker doit être clairement identifié par un numéro de modèle, sa désignation, les fonctionnalités supportées et les connecteurs utilisés.

(4)/S Le pacemaker dispose d'un manuel de fonctionnement et d'un logiciel de configuration, suivi des mesures. Nous ne présentons pas cette partie.
(5)/S Le pacemaker le DCM doit utiliser une télémétrie par induction pour communiquer avec le générateur de pulsation, maintenant une communication à une distance inférieure à 5cm; ou un dispositif ultra-sons ou par radiofréquence dans les limites légales.
(6)/S Des sondes bipolaires placées sur l'atrium et le ventricule doivent être utilisables. L'impédance des sondes doit être comprise entre 100Ω et 2500Ω.
(7)/S Les pulsations générées par le pacemaker doivent être programmable en amplitude et en largeur.
(8)/S La mesure des pulsations doivent se faire à l'aide d'électrode. La décision de détecter un battement doit être basée sur la mesure des longueurs de cycles. La mesure doit s'effectuer sur une fenêtre glissante.
(9)/S La sensitivité des sondes doit être un paramètre ajustable par le médecin. Chaque sonde doit être programmable indépendamment.

Catégorie	Zone contrôlée	Zone mesurée	Réponse à une mesure	Modulation
Lettres	O - Rien	O - Rien	O - Rien	R - Modulation du rythme
	A - Atrium	A - Atrium	T - Déclenché	
	V - Ventricule	V - Ventricule	I - Inhibé	
	D - Double (A + V)	D - Double (A + V)	D - Double (T + I)	

TABLE 3.1 - Mode opérationnel bradycardie d'un pacemaker[20].

Le pacemaker est un dispositif électronique programmable. Son comportement général est invariant, et est défini par le fonctionnement intrinsèque du cœur. En revanche, certaines situations sont ajustables en fonction de la pathologie du patient. Les modes bradycardies énumèrent les différentes situations pour lesquelles le pacemaker peut être programmé.
(10)/S **(Nommage)** Le mode opérationnel d'un pacemaker est défini par un code universel composé de trois ou quatre caractères. Ce code fournit une indication claire sur les fonctions activées à un instant donné. Cette séquence est appelée "mode opérationnel du pacemaker" (voire table 3.3). En pratique, seules les trois ou quatre premières lettres sont utilisées pour décrire les modes dans le cas d'une bradycardie. La première lettre représente la chambre stimulée, la seconde celle qui est mesurée, la troisième pour indiquer si on doit réagir à une mesure and la dernière lettre, optionnelle, indique qu'il faut moduler le rythme cardiaque du fait d'une activité physique, mesurée par l'accéléromètre. 'La lettre 'X" est utilisée pour indiquer toutes les autres lettres (c'est-à-dire "O", "A", "V" or "D"). La lettre "T" ("triggered", déclenché) indique la délivrance d'un stimulus, un signal électrique, tandis que

“I” (“inhibited”, inhibé) indique son absence après avoir mesuré une activité intrinsèque d’une des chambres du cœur.

Les exigences qui suivent définissent le comportement à court terme du pacemaker, c’est à dire son comportement sur une période de temps de l’ordre de quelques battements cardiaques.

(11)/S **(Pas de réponse à une mesure (O))** Stimulation sans mesure, ou stimulation asynchrone correspond à une stimulation délivré sans tenir compte des valeurs mesurées.

(12)/S **(Réponse déclenchée (T))** Dans ce mode, le résultat d’une mesure dans un chambre doit déclencher immédiatement une stimulation dans cette même chambre.

(13)/S **(Réponse inhibée (I))** Dans ce mode, une mesure sur une chambre doit inhiber une stimulation en attente dans cette même chambre.

(14)/S **(Réponse suivie (D))** Dans ce mode, une mesure doit déclencher une stimulation du ventricule après un délai programmé AV, à moins qu’un mouvement de ce même ventricule soit détecté dans l’intervalle.

Le pacemaker définit une série de modes à plus longue échéance, définissant un état courant pour le patient : permanent, temporaire, stimulation immédiate, magnet ?, et remis à l’état initial (POR, Power-On Reset). Ces modes sont mutuellement exclusifs.

(15)/S **(État permanent)** Cet état est le mode de fonctionnement normal du pacemaker. Les paramètres configurés pour le patient doivent être utilisés.

(16)/S **(État temporaire de bradycardie)** Cet état correspond à un jeu de paramètres qui permettent de tester les paramètres du pacemaker, et fournir un diagnostic du patient. Ce mode est terminé par l’une des actions suivantes : une stiumation immédiate, une commande de l’unité DCM.

(17)/S **(Stimulation immédiate)** Une stimulation bradycarde d’urgence commandée (Pace-Now) doit être disponible. Les paramètres sont les suivants :

1) Le mode de la stimulation doit être “VVI” ;
2) La limite basse de la stimulation doit être de 65ppm• ± 8ms ;
3) L’amplitude de la stimulation doit être de 5V ± 0.5V ;
4) La largeur de la stimulation est de 1ms ± 0.02ms ;
5) La période réfractaire ventriculaire est de 320ms ± 8ms ;
6) La sensibilité ventriculaire est de 1.5mV ;
7) La première stimulation doit être émise au plus tôt après deux cycles plus 500ms depuis la dernière activation ;
8) Une fois activé, ce mode doit être maintenu jusqu’à recevoir un contre-ordre du DCM.

(18)/S **(Test de l'aimant)** lors du test de l'aimant

(19)/S **(Remise à l'état initial)** Ce mode est entré lorsque le niveau de la batterie est trop bas pour rendre le comportement du système efficace. Toutes les fonctions doivent être désactivées jusqu'à ce que la batterie atteigne à nouveau un seuil suffisant. Une fois ce seuil atteint, les paramètres du pacemaker sont les suivants :

1) Le mode de la stimulation doit être "VVI" ;

2) La limite basse de la stimulation doit être de 65ppm± 8ms ;

3) L'amplitude de la stimulation doit être de 5V ± 0.5V ;

4) La largeur de la stimulation est de 0.5ms ± 0.02ms ;

5) La période réfractaire ventriculaire est de 320ms ± 8ms ;

(20)/S **(Test Aimant)** Ce mode est utilisé pour déterminer le statut de la batterie. Un aimant standard, ayant la forme d'un donut, doit être détecté par le pacemaker une distance de 2.5cm entre le centre de la surface du pacemaker, et celle de l'aimant.

Lorsque l'aimant est en place, le périphérique doit :

1) Stimuler de manière asynchrone à un rythme fixe. Le mode du pacemaker doit être "AOO" si le mode précédent était "AXXX", "VOO" si le mode précédent était "VXXX", "DOO" si le mode précédent était "DXXX", ou "OOO" si le mode précédent état "OXO" ;

2) Au démarrage du pacemaker, le rythme doit être de 100 battements par secondes; lorsque le niveau de la batterie a baissé jusqu'à un certain seuil, le rythme descend à 90ppm, puis 85ppm, et continue de baisser au fur et à mesure de la baisse du voltage de la batterie.

3) Lorsque l'aimant est retiré, le pacemaker doit prendre pour hypothèse qu'il effectue un pré-test.

4) Ce mode doit pouvoir être désactivé, afin d'ignore la détection de l'aimant.

(21)/S **(Données implantatoires)** Le pacemaker doit pouvoir stocker les informations suivantes en mémoire :

1) modèle de pacemaker, numéro de série, date d'implantation :

2) date d'implantation des sondes, leur polarité ;

3) seuil de stimulation, amplitude des signaux ;

4) impédance des sondes ;

5) paramétrage des signaux, spécifiques au patient.

3.8 Comportement du pacemaker

Dans la section précédente, nous avons présenté les exigences liées au pacemaker, ainsi que ses différents modes de fonctionnement et son interaction avec le patient et l'équipe médicale. Nous avons mis en évidence les différents composants du pacemaker, les paramètres associés. Dans cette section, nous revenons sur le pacemaker, en étudiant son comportement temporel dans le cas de la thérapie de la bradycardie.

Le traitement de la bradycardie s'opère en stimulant le cœur du patient. Ce traitement doit être adapté au patient. Il y a donc plusieurs paramètres qui sont utilisés en fonction du mode de stimulation et des besoins du patient.

Nous définissions tout d'abord plusieurs termes qui seront utiles pour définir le comportement du pacemaker en réaction à une stimulation cardiaque, ou son absence passée un certain temps.

(22)/P **(Limite basse de stimulation)** La limite basse de stimulation (LRL) est le nombre de stimulations produites par minute (dans le cas atrial ou ventriculaire) en l'absence d'activité intrinsèque du cœur et qu'il n'y a pas de stimulations contrôlées par le capteur à une fréquence plus élevée.

La définition du LRL varie suivant le contexte :

– Lorsque l'hystérésis est désactivée, la LRL définit la plus longue période autorisée entre deux stimulations ;

– Dans les modes DXX ou VXX, la LRL démarre lorsqu'une pulsation du ventricule est détectée, ou qu'une stimulation est détectée ;

– Dans les modes AXX, la LRL démarre lorsqu'une pulsation de l'oreillette est détectée, ou qu'une stimulation est détectée.

(23)/P **(Limite haute de stimulation)** La limite haute de pulsation (URL) est le nombre maximum d'événements mesurés au niveau de l'oreillette. L'URL est l'intervalle minimum entre un événement du ventricule, et la prochaine stimulation du ventricule.

(24)/P **(Délai d'attente atrio-ventriculaire)** Le délai attente atrio-ventriculaire (AV) est un paramètre programmable représentant l'intervalle entre un événement atrial (intrinsèque ou stimulé) et une pulsation ventriculaire.

Dans les modes de suivi de l'oreillette, la stimulation ventriculaire doit se produire en absence d'un événement détecté du ventricule au plus tard après ce délai d'attente lorsque le taux de pulsation de l'oreillette est compris entre LRL et URL. Le délai d'attente peut être fixé (temps absolu) ou relatif. Ce délai d'attente a une valeur différente suivant l'événement déclencheur :

(25)/P **(Délai d'attente atrio-ventriculaire stimulé)** Le délai d'attente stimulé doit avoir lieu lorsque le délai AV celui-ci est initié par une stimulation de l'oreillette.
(26)/P **(Délai d'attente atrio-ventriculaire détecté)** Le délai d'attente détecté doit avoir lieu lorsque le délai AV celui-ci est initié par une pulsation de l'oreillette.
(27)/P **(Délai d'attente atrio-ventriculaire dynamique)** Dans le cas où le délai d'attente est dynamique, sa valeur est déterminée pour chaque nouveau cycle cardiaque, en fonction des cycles précédents. Ce délai d'attente est obtenu en multipliant la durée du cycle précédent par une constant stockée en mémoire. Sa valeur est bornée par une valeur minimum et une valeur maximum.
(28)/P **Décalage du délai d'attente** Le paramètre configurable de décalage permet de réduire le délai d'attente après une détection d'une pulsation suivie au niveau de l'oreillette.
(29)/P **Période réfractaire** Afin de limiter les erreurs de détection, une période réfractaire pendant laquelle les mesures des sondes sont ignorées peut être mise en place. Différentes périodes réfractaires sont programmables, après un événement ventriculaire (VRP), atrial (ARP) et post atrio-ventriculaire (PVARP). Dans ce dernier cas, la période réfractaire est l'intervalle de temps suivant un événement ventriculaire lorsqu'un événement au niveau de l'oreillette n'inhibe pas une stimulation spontanée de l'oreillette ni ne déclenche une stimulation ventriculaire.
La période PVARP étendue est définie comme suit : si ce paramètre est actif, tout occurrence prématurée d'une contraction du ventricule (PVC) force une période réfractaire du générateur de pulsation pour une durée égale à ce paramètre ; la valeur de la période réfractaire doit revenir à la normal au cycle cardiaque suivant.
(30)/P **Réponse au bruit** En cas d'un bruit continu, le pacemaker doit être en mode de stimulation asynchrone.
(31)/P **Réponse à une tachycardie atriale** La réponse à une tachycardie atriale (ATR) vise à éviter un rythme cardiaque soutenu pendant de longues durées. Dans ce mode, le générateur de pulsations doit détecter une situation de tachycardie si le rythme intrinsèque de l'oreillette dépasse la valeur URL pendant un certain temps.
(32)/P **Détection de tachycardie** La détection de tachycardie fonctionne comme suit :
– Le début d'une situation de tachycardie est détectée lorsque le rythme cardiaque est en moyenne supérieur à URL ;
– La fin d'une situation de tachycardie est détectée lorsque le rythme cardiaque est en moyenne inférieur à URL ;

– La période de détection doit être suffisamment courte pour que la thérapie se mette en place rapidement ;
– La période de détection doit être suffisamment longue pour que la thérapie ne se déclenche pas trop souvent, ni inutilement.
(33)/P **ATR en cours** Lorsqu'une situation de tachycardie est détectée, l'algorithme de l'ATR entre dans l'état "ATR en cours". Le générateur de pulsations doit attendre un nombre de cycles cardiaques programmés avant d'entrer dans l'état "repli" (fallback). L'algorithme sort immédiatement de cet état lorsque l'attente est terminée, ou que la situation de tachycardie prend fin.
(34)/P **Repli ATR** Si la situation de tachycardie perdure après le délai d'attente précédent, alors (1) le générateur de pulsations entre dans l'état "Fallback" et passe en mode VVIR; (2) le rythme de pulsation est réduits à LRL ; (3) ce mode prend fin dès que la situation de tachycardie cesse ; (4) le changement de mode est synchronisé sur une pulsation du ventricule, ou un événement détecté.
(35)/P **Adaptation du taux de stimulation** Le pacemaker doit pouvoir ajuster le cycle cardiaque en fonction des besoins du métabolisme tels que mesurés par l'accéléromètre.
Les paramètres suivants sont définis :
(36)/P **Taux maximal "capteur"** Le taux maximal "capteur" correspond à la valeur maximum du taux de stimulation en réponse aux mesures du capteur. (1) ce paramètre est requis pour les modes adaptatifs, et (2) programmable indépendamment de l'URL.
(37)/P **Seuil d'activité** Ce seuil correspond au minimum d'activité détecté par l'accéléromètre avant une adaptation de la stimulation cardiaque.
(38)/P **Facteur de réponse** L'accéléromètre doit déterminer le nouveau taux de stimulation en fonction de l'état d'activité du patient. Plus forte est la valeur mesurée, plus rapide sera la réaction et la modification du taux de stimulation.
(39)/P **Temps de réaction** L'accéléromètre doit déterminer le taux d'augmentation du rythme de stimulation. Le temps de réaction correspond à la période pour passer de la valeur LRL à MSR.
(40)/P **Temps de recouvrement** L'accéléromètre doit déterminer le taux de recouvrement correspond à la période pour passer de la valeur MSR à LRL, jusqu'à ce que l'activité passe sous le seuil d'activité.
(41)/P **Stimulation en hystérésis** Lorsque de mode est activé, les stimulations du générateur sont retardées afin de permettre une stimulation spontanée du cœur. Ce mode est actif lorsque le mode de stimulation est inhibé ou suivi, le rythme cardiaque est supérieur au taux limite

d'hystérésis (HRL) ; lorsque le pacemaker est dans le mode AAI, un événement détecté au niveau de l'oreillette doit activer l'hystérésis ; ou enfin dans le mode inhibé et suivi avec stimulation du ventricule, un événement est détecté au niveau du ventricule.
(42)/P **Lissage du rythme** Les variations du rythme de stimulation sont lissées afin de ne pas être trop brusques. Ces paramètres sont donnés indépendamment pour l'augmentation et la diminution du taux.

3.9 Modélisation du pacemaker

Nous avons vu précédemment les principaux sous-systèmes du pacemaker, leurs rôles et leurs interactions :

· Un sous-système d'électrodes (lead dans la spécification) détecte les battements naturels du cœur et stimule le cœur sous l'impulsion du générateur de pulsation. Ce sous-système fournis donc les entrées (sous forme de signaux électriques) du générateur de pulsation, et met en œuvre les sorties (sous forme d'impulsions électriques transmises au cœur) du générateur de pulsation. Notons que deux électrodes sont implantées, une par compartiment du cœur (oreillette et ventricule).

· Un générateur de pulsation (pulse generator - PG - dans la spécification) reçoit les signaux électriques délivrés par les électrodes de mesure de l'activité cardiaque naturelle, et émet des signaux électriques vers les électrodes de stimulation pour déclencher les battements du cœur (dans le compartiment concerné). Comme nous allons expliquer par la suite, ce sous système présente des modes de fonctionnements auxquels sont associées des fonctionnalités de natures différentes, et des contraintes temporelles différentes.

· Une station de configuration et de supervision (Device Controller-Monitor dans la spécification - DCM -, manipulée par un praticien (médecin, infirmier, etc…) pour configurer le générateur de pulsation et/ou superviser le comportement de ce dernier (en demandant par exemple les données nécessaire pour produire un histogramme de l'activité cardiaque).

Le schéma général du système et le suivant:

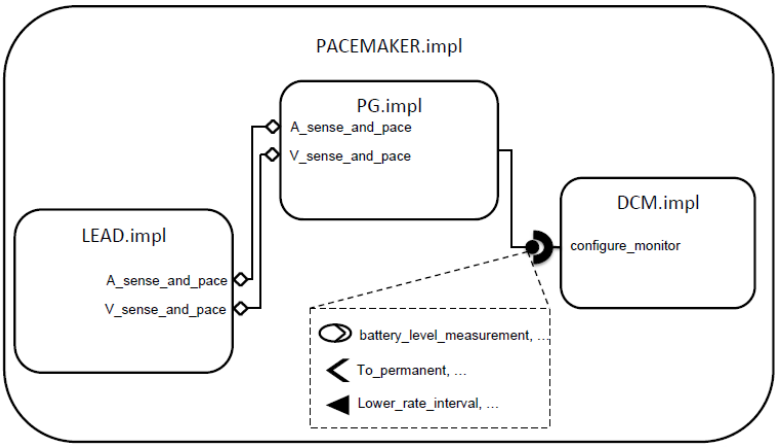

FIGURE 3.3 – Le système global

Parie logicielle

Le processus de contrôle du PG est le suivant:

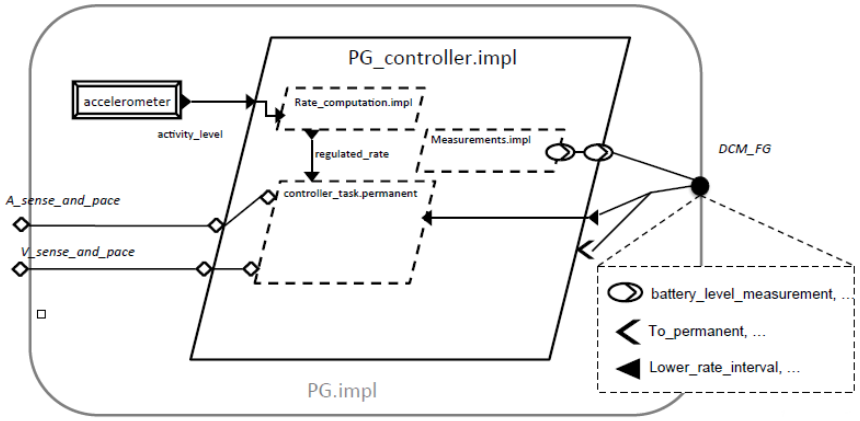

FIGURE 3.4 -Processus PG

Le sous-système LEAD est constitué de deux Capteurs (ventriculaire/atriale) des battements naturels. Le schéma suivant illustre cette composition.

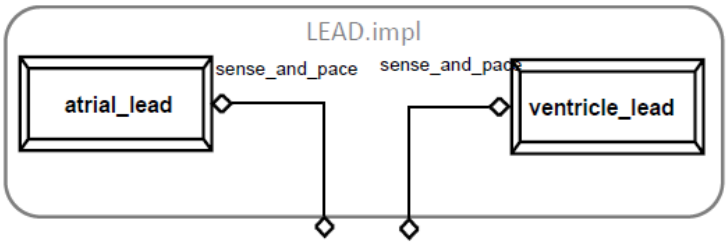

FIGURE 3.5 -Le sous-système LEAD

Le sous-système PG (Pulse Generator) est une ressource d'exécution des fonctionnalités de régulation cardiaque.

Le sous-système DCM est une ressource d'exécution des fonctionnalités de régulation cardiaque.

Le schéma de communication entre le PG et le DCM est le suivant.

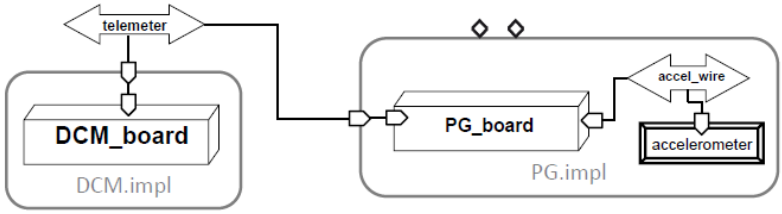

FIGURE 3.6 - Communication PG - DCM

3.10 Conclusion

Dans ce chapitre, nous avons retenu comme cas d'étude un pacemaker, un système électrique dont le rôle est de stimuler le cœur d'un patient atteint de bradycardie. Nous avons détaillé la structure du pacemaker et son fonctionnement en présentant l'architecture générale du pacemaker et les exigences auxquelles il doit répondre.

4.1 Introduction

Pour faire une modélisation du pacemaker on ~~en~~ peut utiliser plusieurs outils parmi ces outils on trouve Atom3 et Dia donc dans ce chapitre, nous présentons sommairement ces outils et comment les utiliser avec des exemples.

4.2 Dia

4.2.1 Définition :

Dia est un programme de création de diagrammes basé sur gtk+. Il est distribué sous licence GPL. dia est conçu pour ressembler au programme commercial sous Windows nommé "Visio". On peut l'utiliser pour dessiner de nombreuses sortes de schémas différents. Il contient des objets spéciaux pour faciliter le dessin de schémas relationnels, diagrammes UML, organigrammes, plans de réseaux et de circuits simples. Il est aussi possible d'ajouter de nouvelles formes en écrivant de simples fichiers XML, en utilisant un sous-ensemble de SVG pour dessiner les formes.

Dia peut charger et enregistrer des schémas au format standard XML (compressés -gzip- par défaut pour économiser l'espace), peut exporter aux formats EPS ou SVG et peut imprimer les schémas, même ceux qui s'étendent sur plusieurs pages.

4.2.2 Notions de base :

a. Démarrage

Une fois installée, l'application se lance par :

Dia Cela donne :

Figure 4.1 Quelques outils Dia

Au premier lancement, nous choisissons File...Préférences (Fichier... Préférences) et nous remplissons selon nos choix personnels. Ces choix sont sauvegardés dans notre répertoire home sous le nom .dia/diarc. Par exemple, nous utilisons en général les choix par défaut en m'assurant des réglages suivants :

- ✓ User Interface... (Interface utilisateur...)
- ✓ Compress...no (Compression... non) Ainsi je peux facilement manier les modèles UML ensuite. Si j'avais des soucis d'espace disque, j'utiliserais "yes" (oui)
- ✓ Gridlines... (Lignes de grilles...)
- ✓ Visible...yes (Visibles... oui)
- ✓ Snap to...yes (Coller à la grille... oui)

Après avoir cliqué sur le bouton OK dans les préférences, vous pouvez fabriquer un schéma.

b. Nouveau diagramme

Cliquer sur File...New (Fichier... Nouveau) donne ceci :

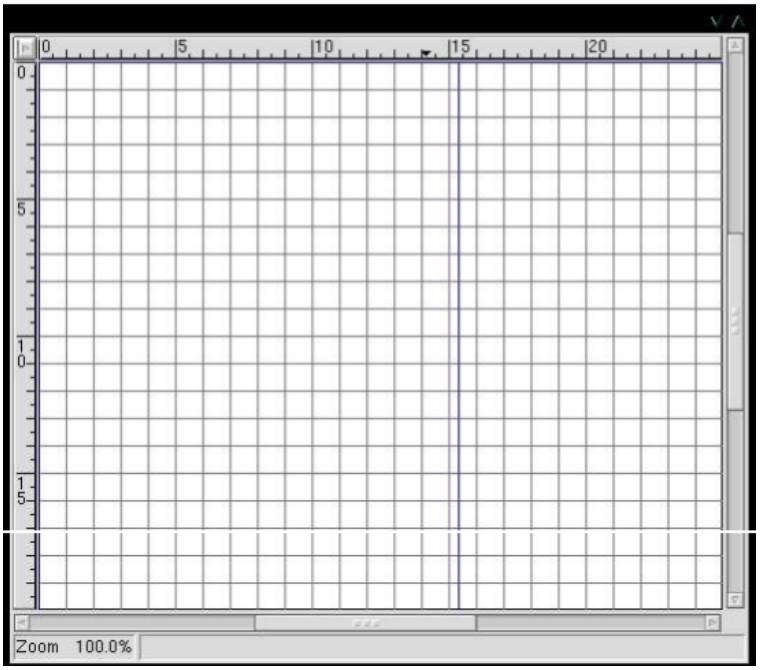

Figure 4.2 l'interface de dia

Un clic droit dans cette fenêtre donne un menu contextuel spécifique aux schémas.

Depuis celui-ci nous choisissons File...page setup (Fichier... Mise en page)

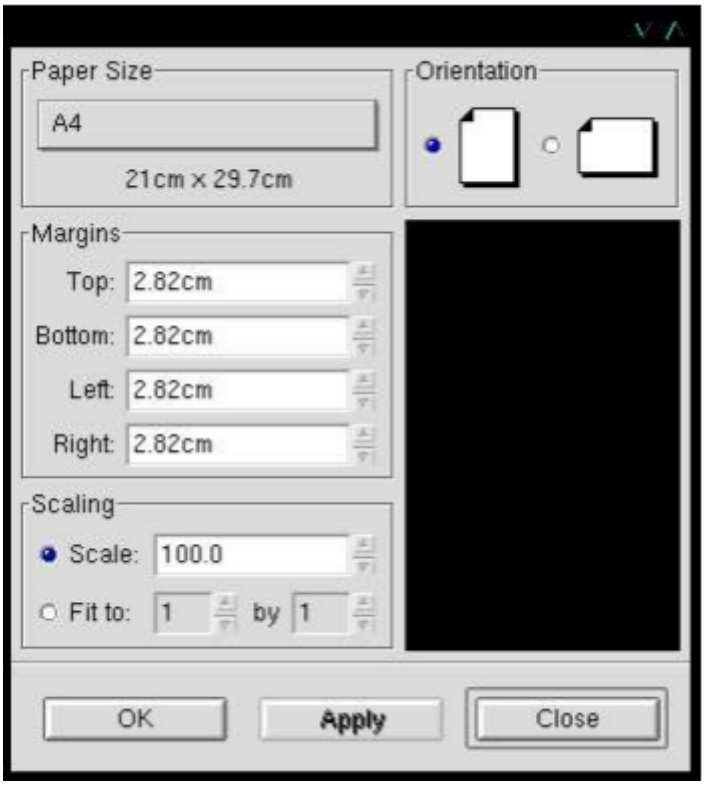

Figure 4.3 Organisation de la feuille de travail dans Dia

Classiquement :

- ✓ Paper size (taille papier) : utiliser A4 pour le papier 21x29,7.

- ✓ Orientation : utiliser portrait pour la plupart des pages web, et paysage pour des affiches.
- ✓ Margins (marges) : Laisser comme c'est. Leur taille s'adapte selon la taille du papier.
- ✓ Scaling (échelle) : Laisser à 100. Nous reviendrons sur ce point plus tard.

Après avoir cliqué sur OK, nous sommes prêts à construire le diagramme.

c. Dessiner un organigramme

Un organigramme nécessite diverses boîtes avec du texte, des lignes pleines et quelques lignes pointillées. On peut construire cela avec les outils de base. Voyons comment s'en servir. Voici la structure de la fenêtre principale :

Figure 4.4 Fenêtre principale de dessin

d. La façon la plus simple de travailler est de :

a. Sélectionner une forme

b. Cliquer pour créer à peu près le nombre de formes dont vous aurez besoin dans votre schéma

c. Les nommer (la plupart des formes acceptent des étiquettes de texte)

d. Déplacer les formes à leur bonne place

e. Les interconnecter avec des lignes. Assurez-vous que les lignes soient bien reliées aux poignées des objets

f. Nommer les lignes si besoin.

g. Déplacer l'ensemble réalisé où vous voulez. Ajouter des points de connexion si nécessaire pour faire des lignes propres.

h. Ajouter/modifier d'autres formes et lignes selon les besoins.

1.3 Cas du pace maker dans Dia

On a ce diagramme qui représente les composants matériels d'un pacemaker

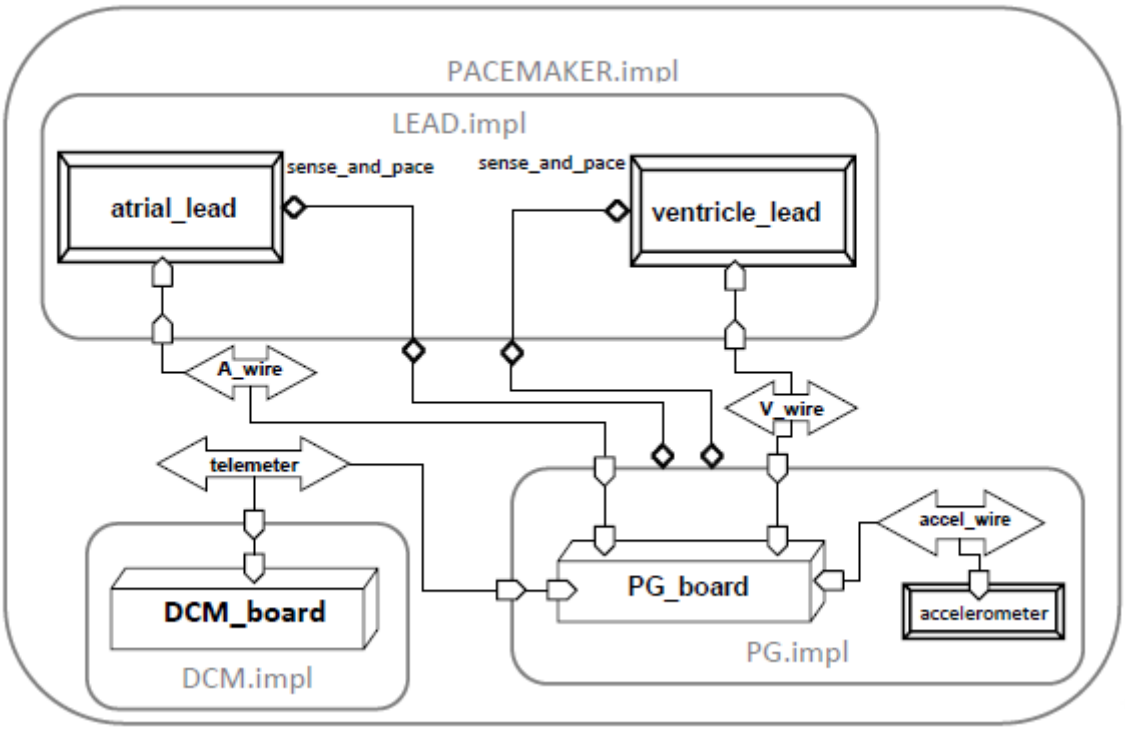

Figure 4.5 les composants matériels d'un pacemaker

On peut le modéliser par Dia comme il est montré dans l'image ci-dessous

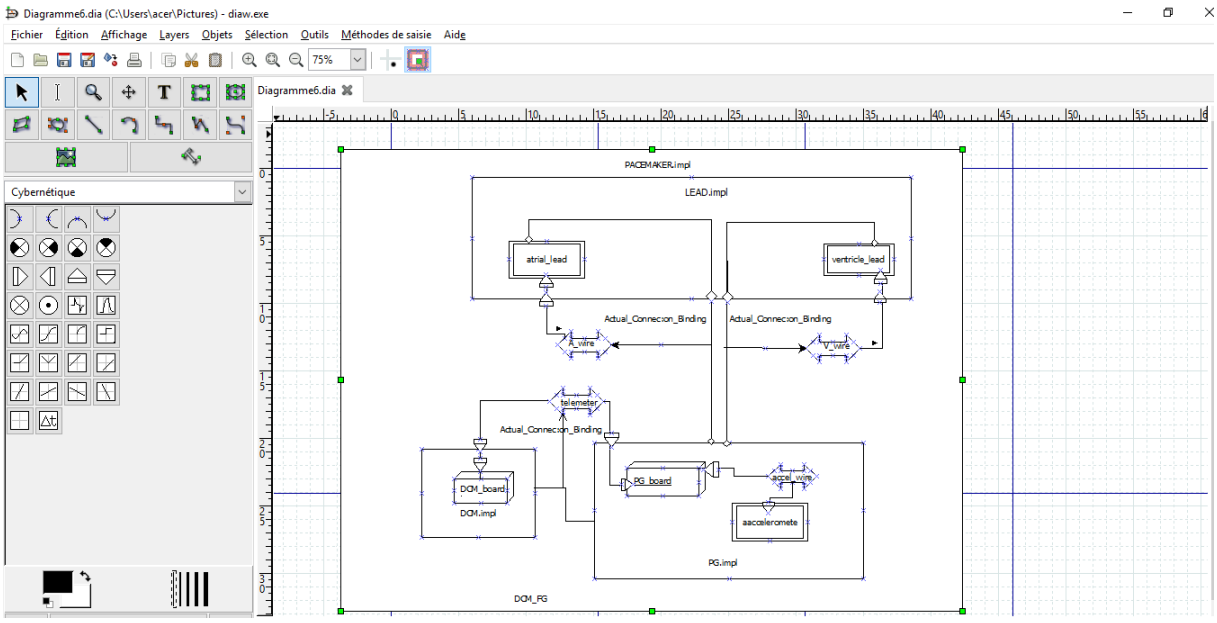

Figure 4.6 modélisation des composants matériels d'un pacemaker

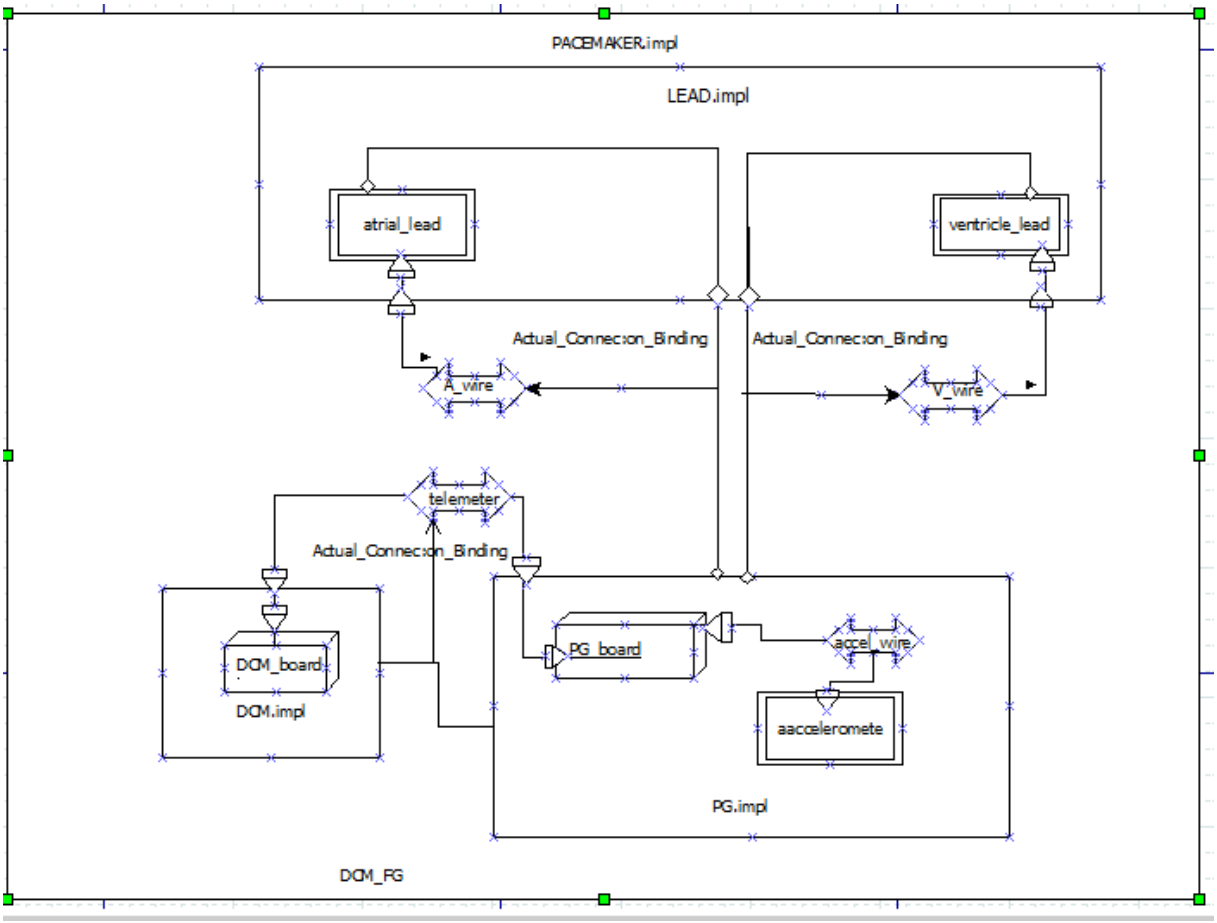

Figure 4.7 modélisation des composants matériels d'un pacemaker plus proche

On peut donner un autre exemple, Le diagramme suivant représente le système global

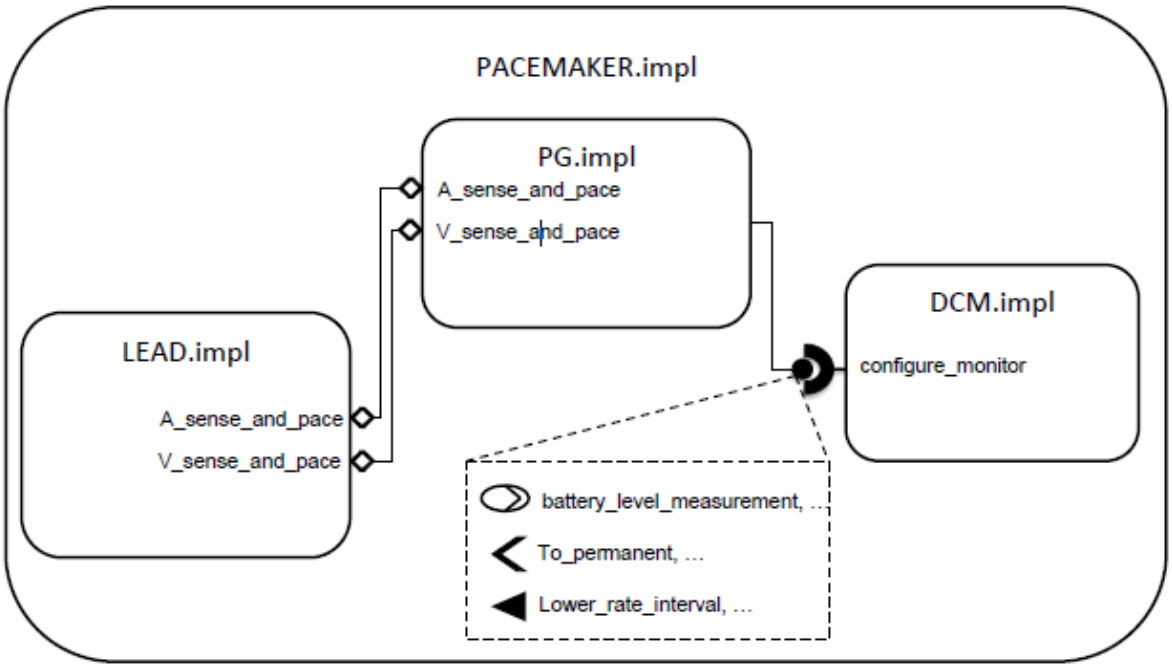

Figure 4.8 Diagramme représentant le système global

Après la modélisation par dia nous obtenons le résultat suivant:

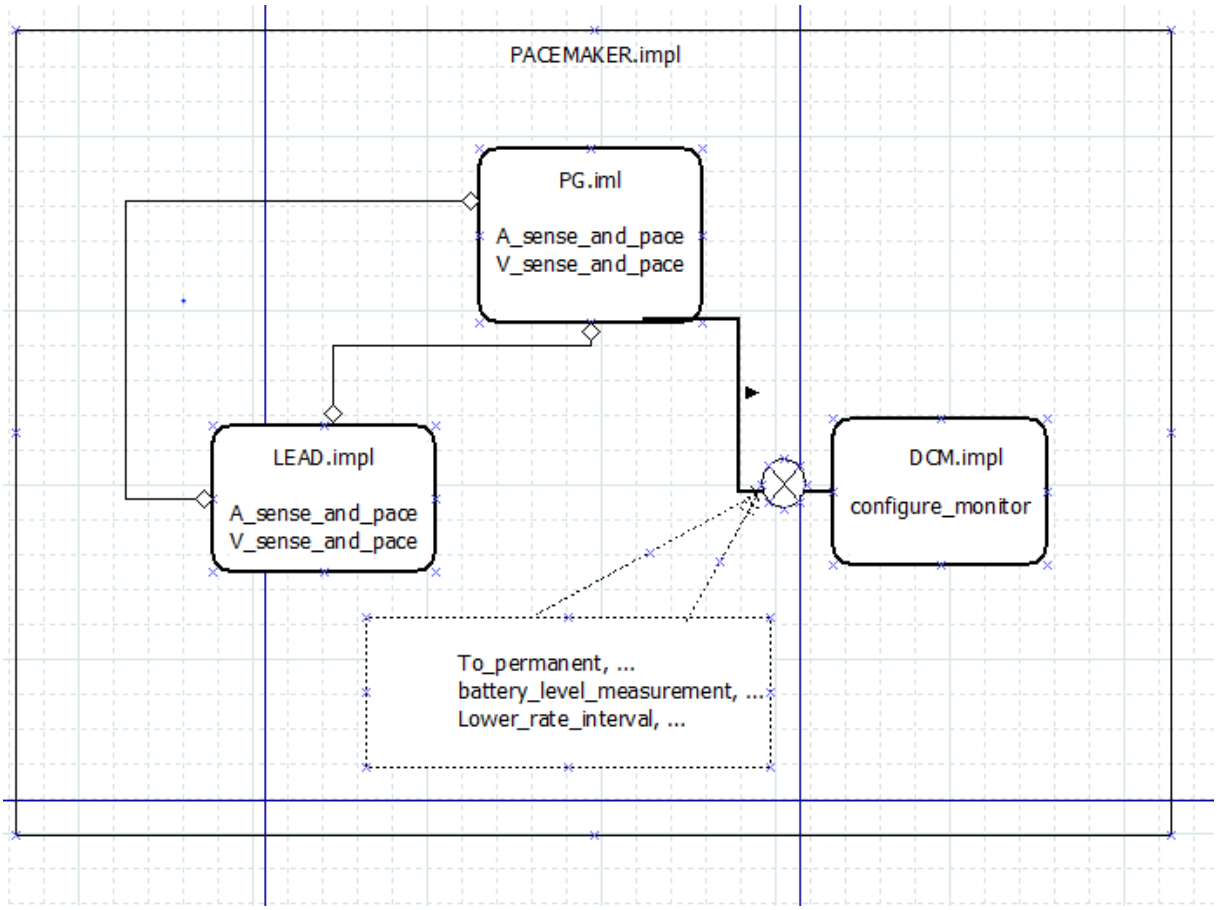

Figure 4.9 la modélisation du système global dans dia

4.3 AToM3 :

4.3.1 Présentation de l'outil AToM3

AToM3 (A Tool for Multi-formalism and Meta-Modelling en anglais) est un outil visuel(optique) de modélisation multi-paradigmes (multiformalisme en anglais) , de manipulation graphique de modèles, Ce outil écrit en Python , et s'exécute sur différentes plateformes (Windows, Linux, etc) développé par le laboratoire MSDL (Modelling, Simulation and Design Lab) à l'université de McGill Montréal, Canada. Cet outil a été conçu en collaboration avec Juan de Lara de l'université de Madrid (UAM), Espagne.

Il est utilisé pour la modélisation, la méta-modélisation et la transformation de modèles à l'aide de grammaires de graphes. Le composant principal d'AToM3 est le noyau (Kernel), qui est responsable du chargement, de la sauvegarde, de la création et de la manipulation des

modèles. Il supporte la méta-modélisation. Les méta-modèles sont exprimés en utilisant le formalisme Entité-Association [30].

Voici l'interface générale d'Atom3

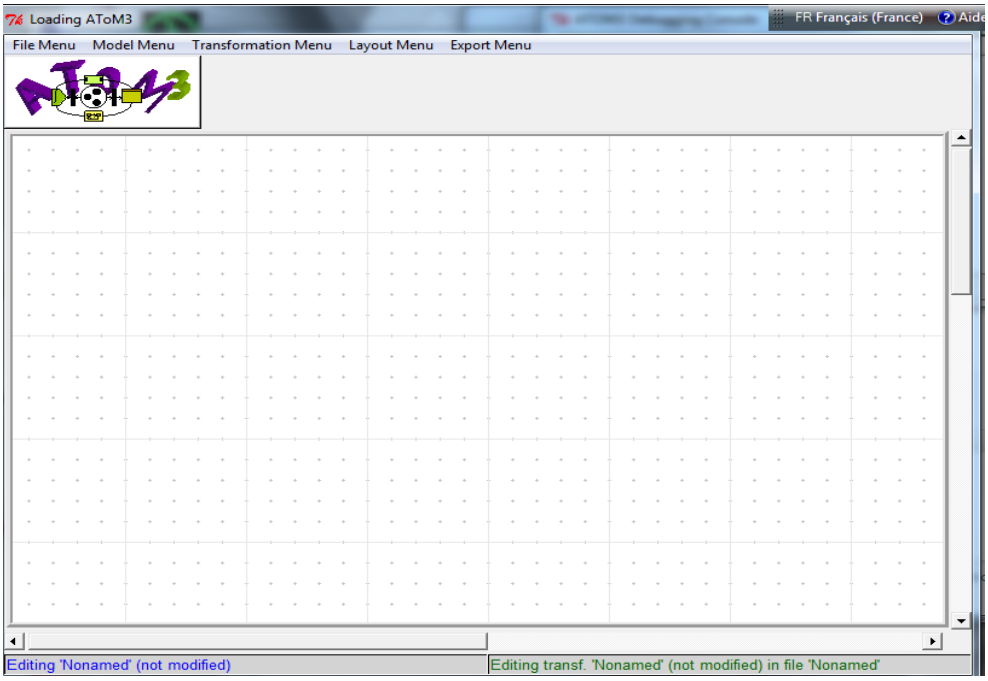

Figure 4.10 Interface d'AToM3.

Pour afficher la barre d'outils nous pressons sur file menu puis en choisissons central formalismDir(voir la figure)

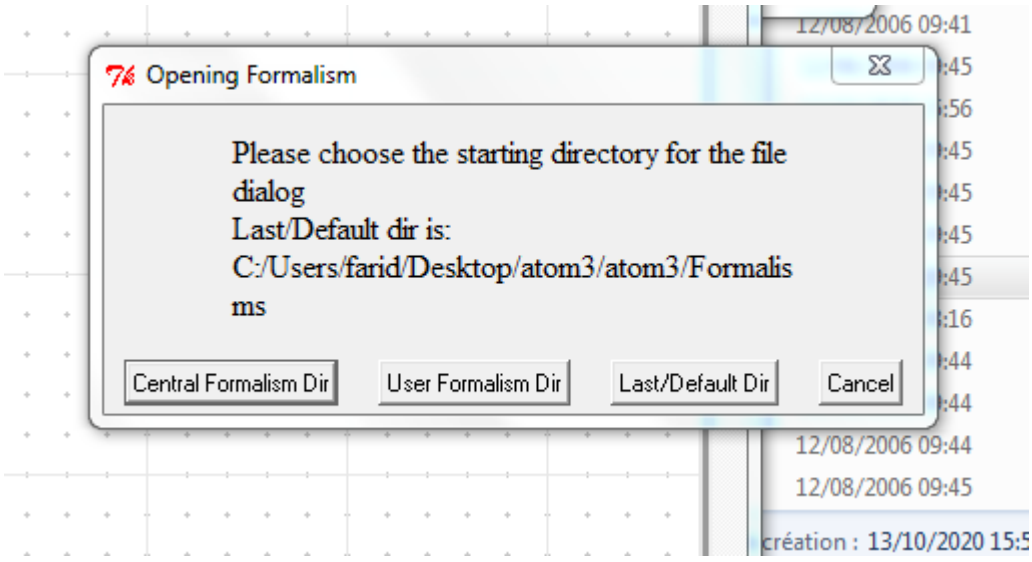

Figure 4.11 étape 1 pour créer un model dans atom3.

Puis, sur la fenêtre qui apparait, on choisit 'CD_classDiagramsV3' comme le montre l'image suivante:

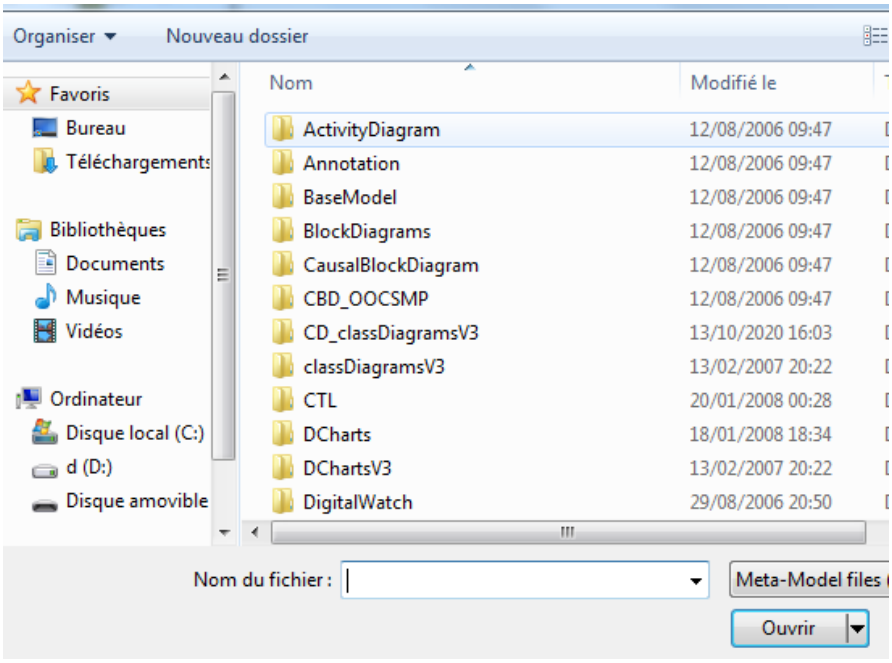

Figure 4.12 étape 2 pour créer un model dans atom3.

Ensuit nous choisissons CD_ClassDiagramsV3_META comme illustré dans l'image au dessous

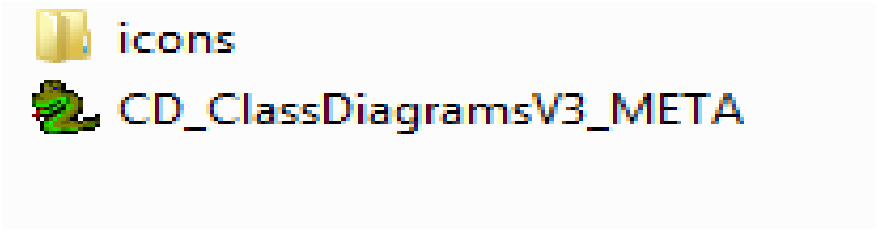

Figure 4.13 étape 3 pour créer un modèle dans atom3.

Finalement nous obtenir même premier interface mais avec la barre d'outille qui permet a nous de réaliser notre modélisation

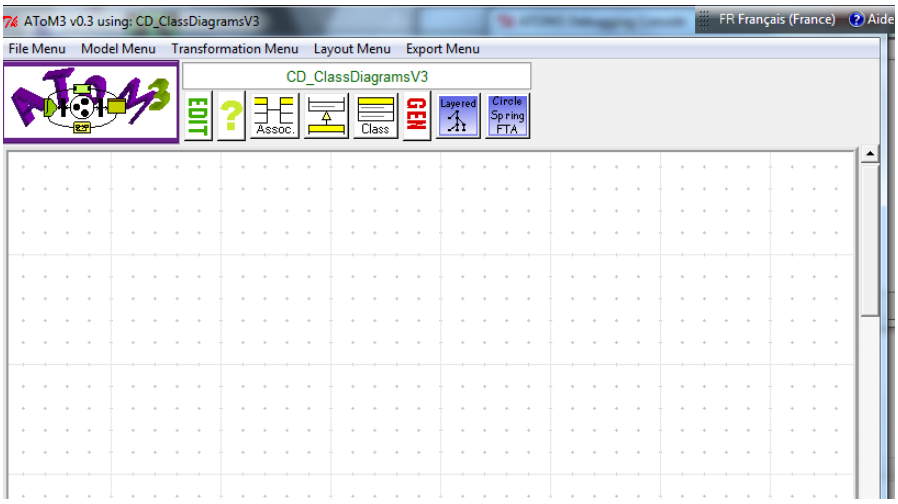

Figure 4.14 atom3 avec barre d'outils

4.3.2 Exemples de cas du pacemaker dans AtoM3

Figure 4.15 contraintes temporelles1

Nous construisons un diagramme de classe en utilisant atom3.

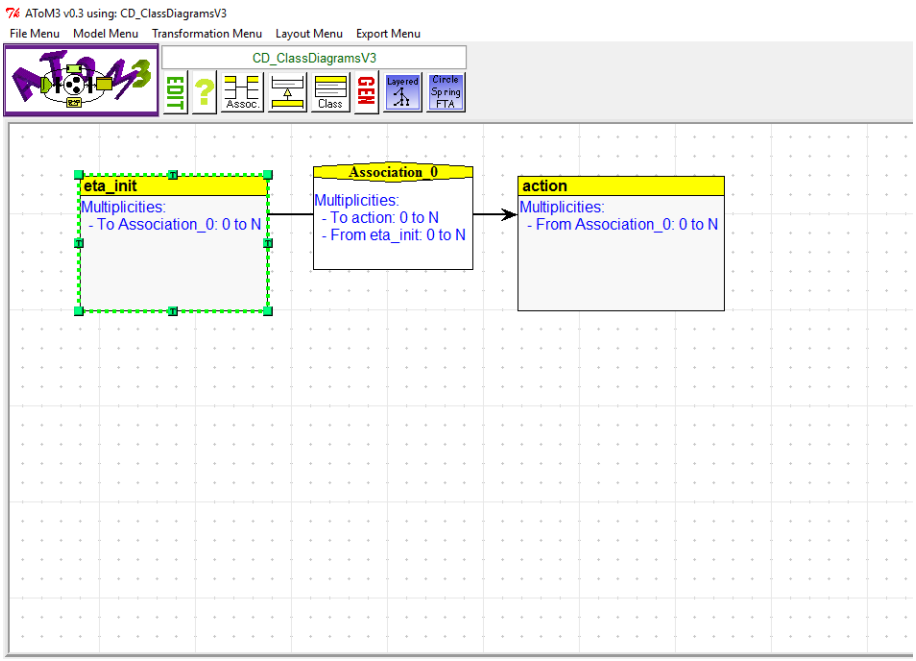

Figure 4.16 diagramme1 de classe dans atom3

Après nous sauvegardons le modèle et nous commençons à faire le dessin en utilisant le bouton ‘Graphical_Appearance‘

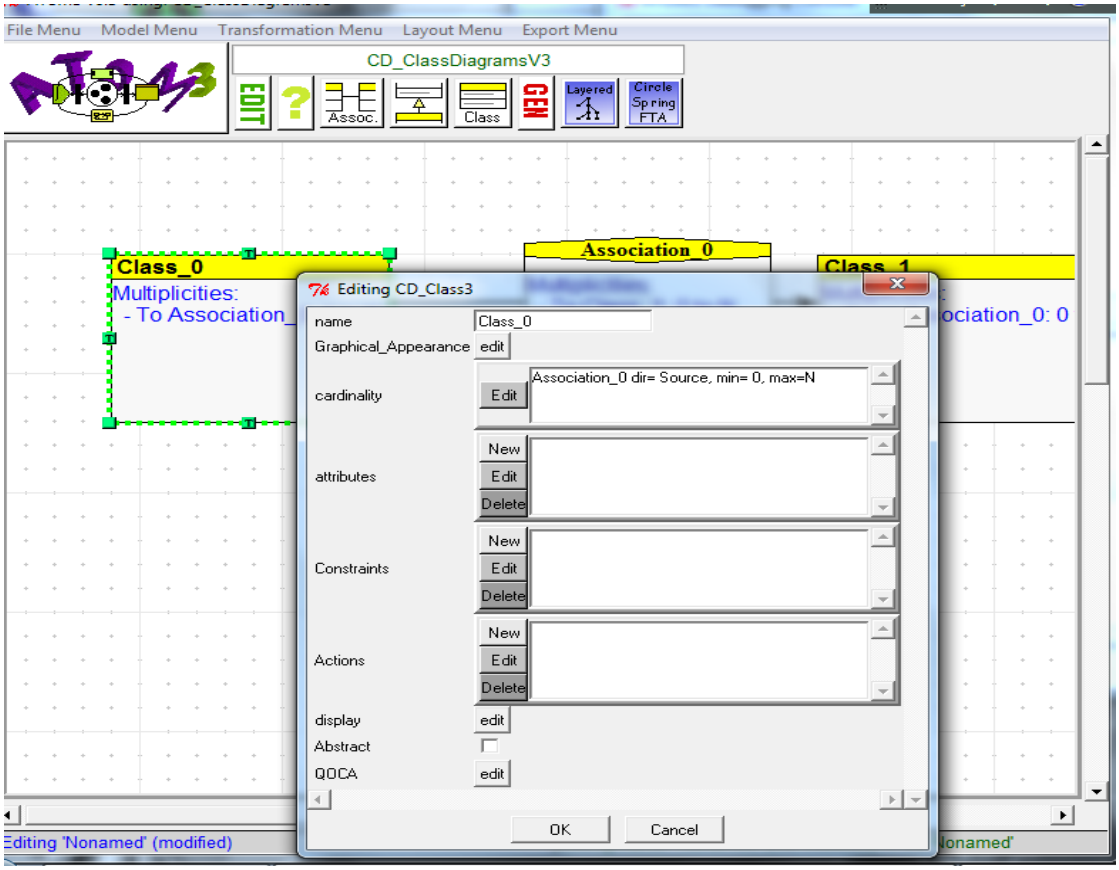

Figure 4.17 le bouton ‘Graphical_Appearance’ dans atom3

Puis nous passons à la fenêtre suivante :

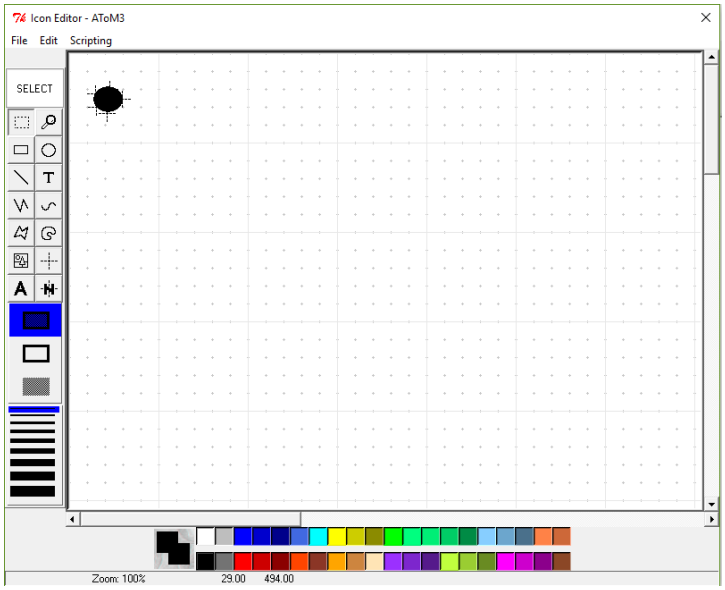

Figure 4.18 désignation de l'état à partir de l’éditeur

Nous mettons des connecteurs (sing +) puis nous pressons ‘Save and exit’

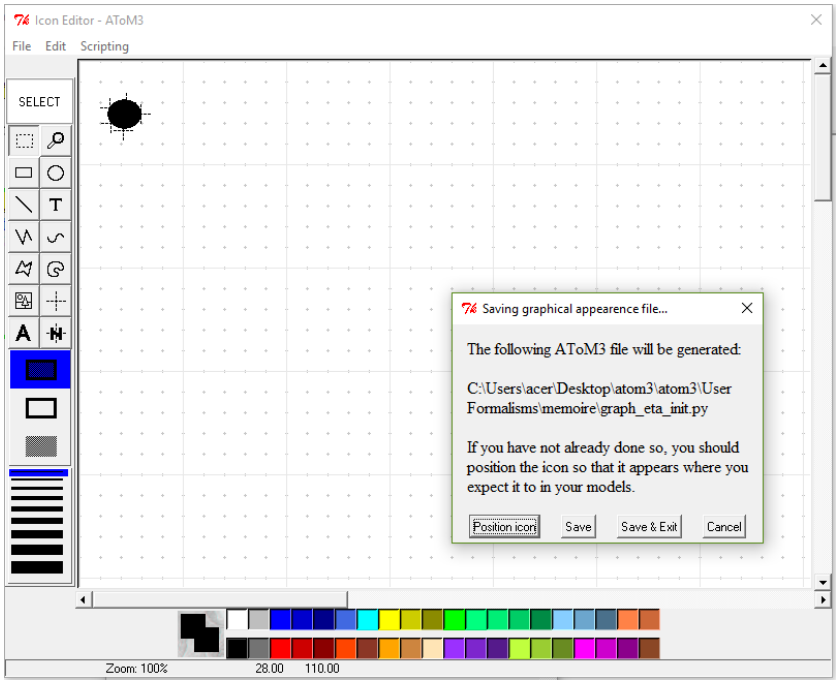

Figure 4.19 Sauvegarde du dessin

Même chose pour la deuxième classe

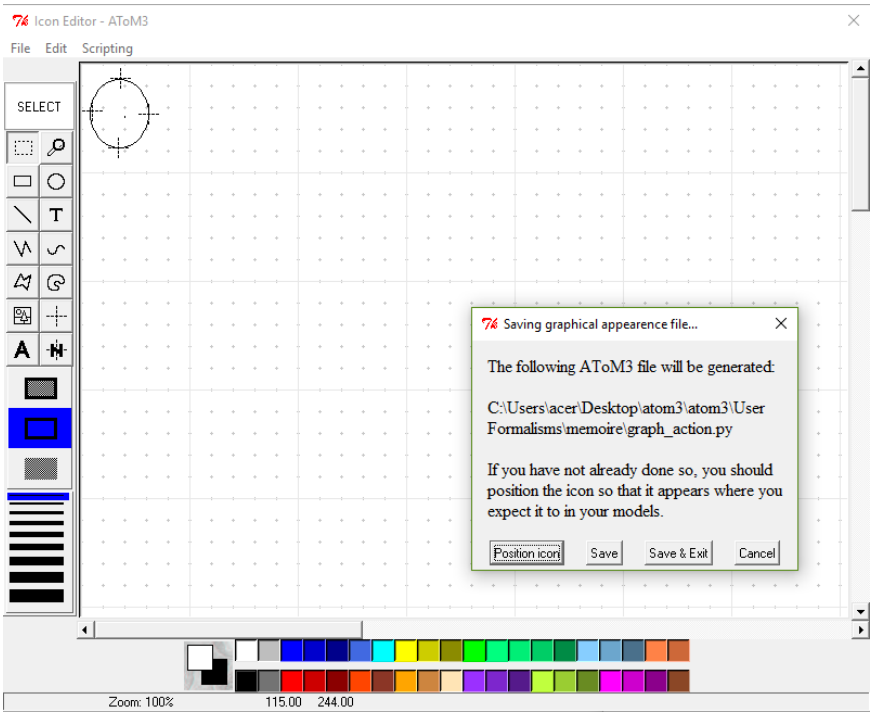

Figure 4.20 création d'état final1

Après nous devons générer le modèle.

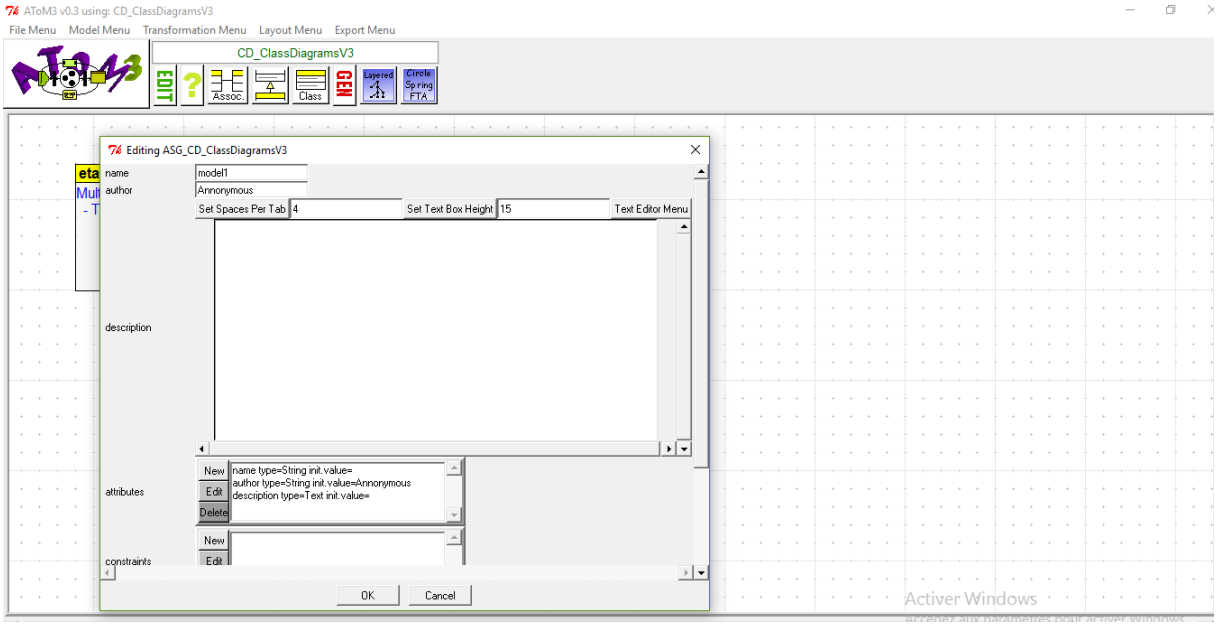

Figure 4.21 génération du modèle1

Dans cette fenêtre il faut préciser les zones suivantes :

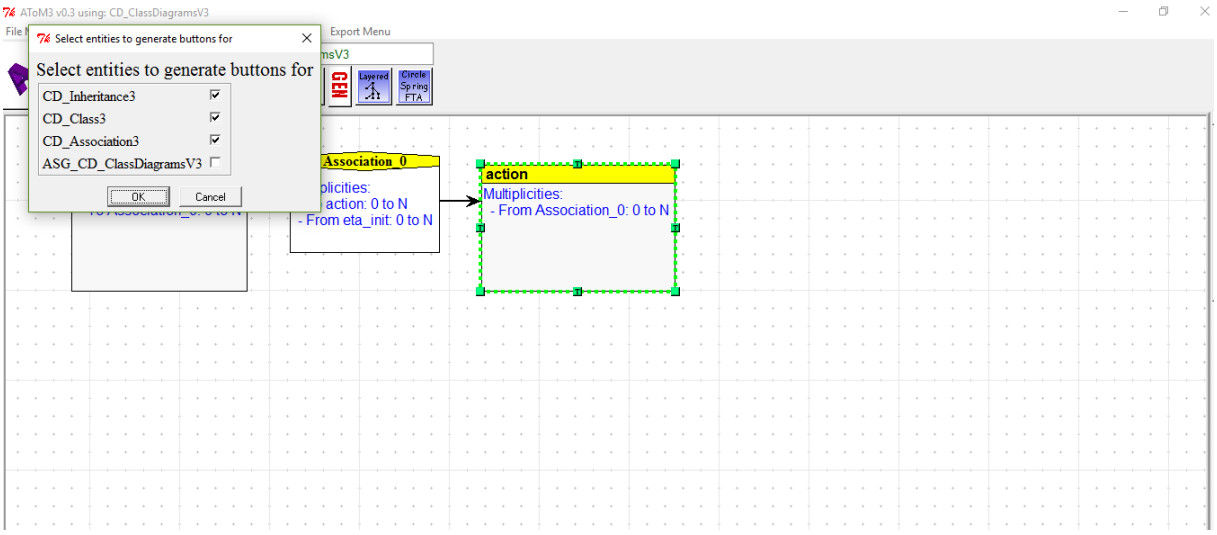

Figure 4.22 suite de génération1

Puis redémarrer atom3 pour obtenir le méta modèle de ce modèle.

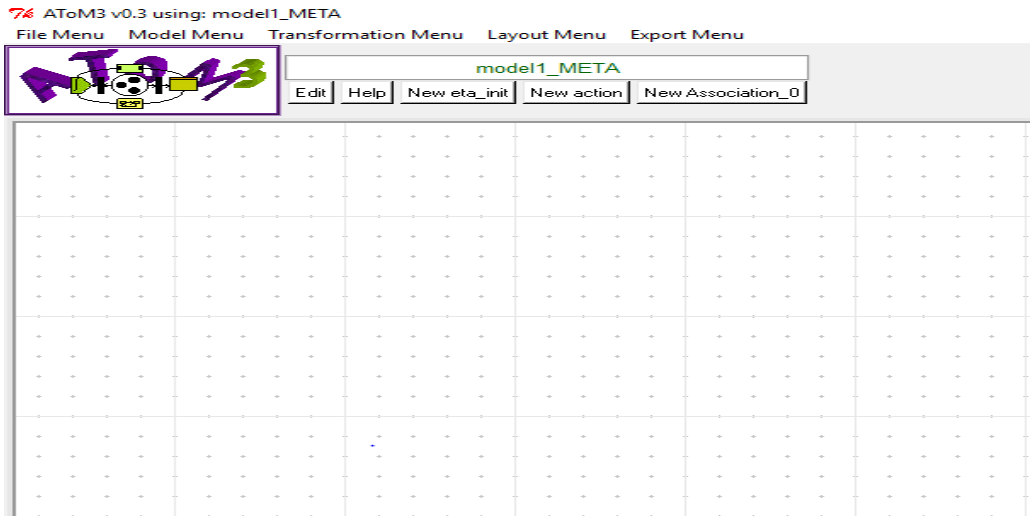

Figure 4.23 – Le méta modèle du diagramme1

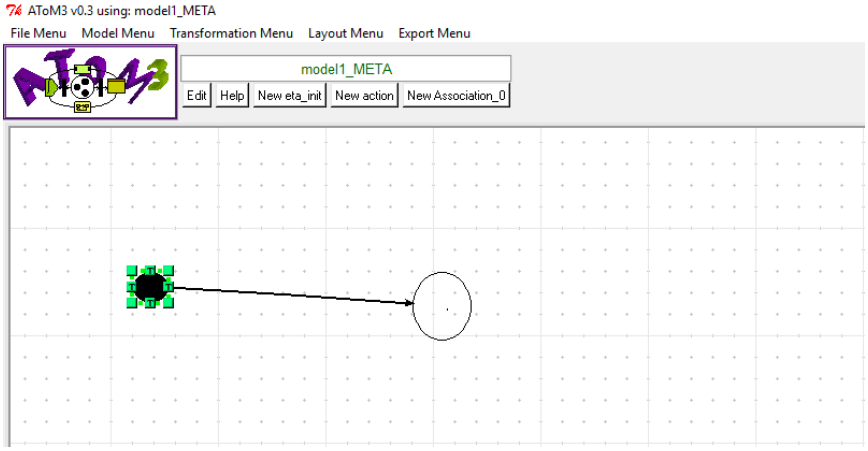

Figure 4.24 – Faire une relation entre l'état initial et l'action

Nous pouvons faire la même modélisation pour des ~~les~~ diagrammes plus complexes comme celui associé aux contraintes temporelle suivantes :

Contraintes temporelles:

- LRL, AV, URL, et rate_value

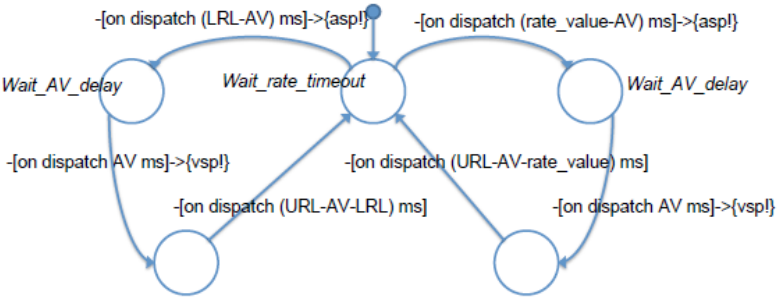

FIGURE 4.25 – Contraintes temporelles2

Comme dans le cas du diagramme précédent, on commence par le diagramme de classe

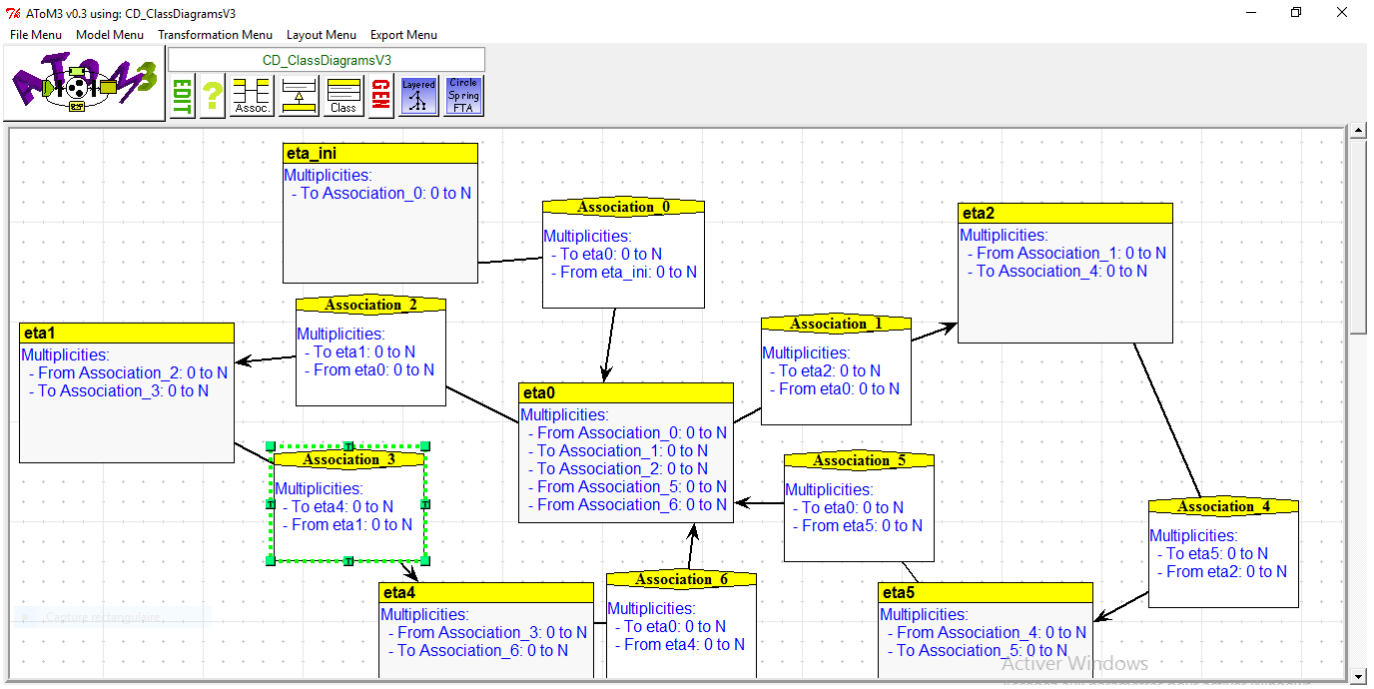

Figure 4.26 – Diagramme2 de classe de Contraintes temporelles précédentes

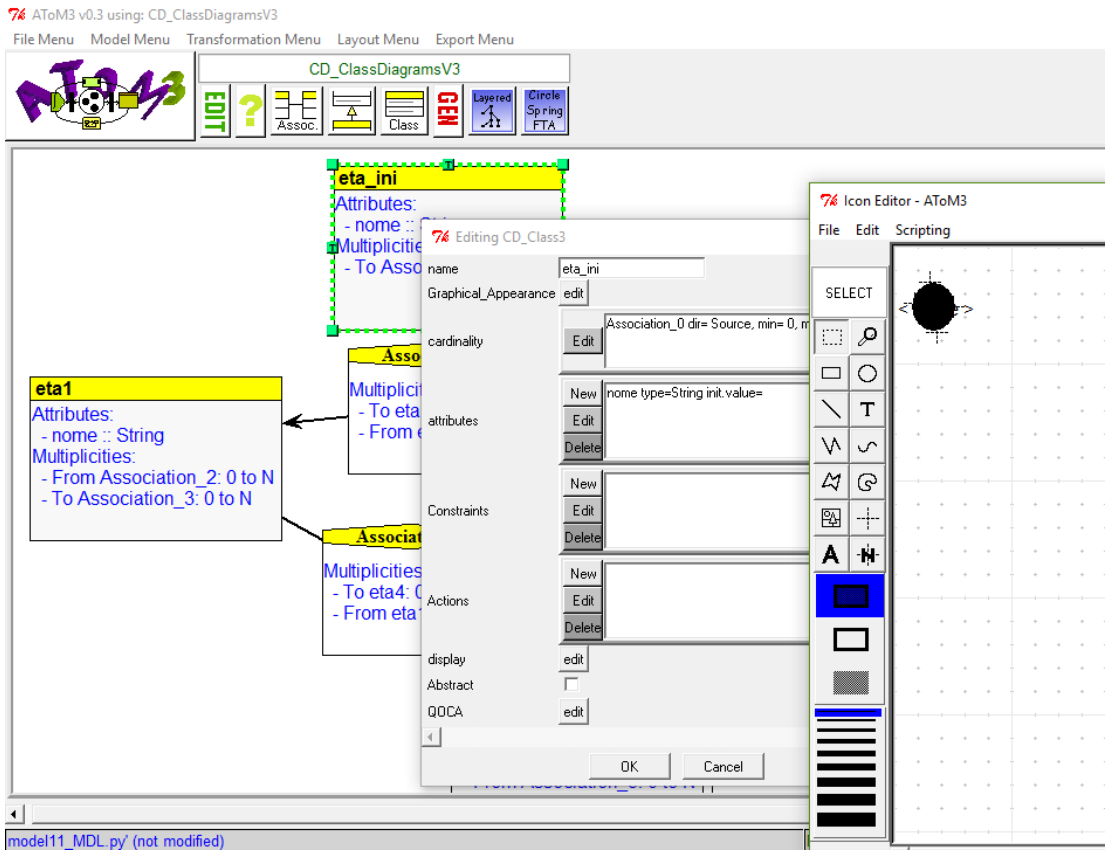

Figure 4.27 –Editeur graphique de la classe initiale2

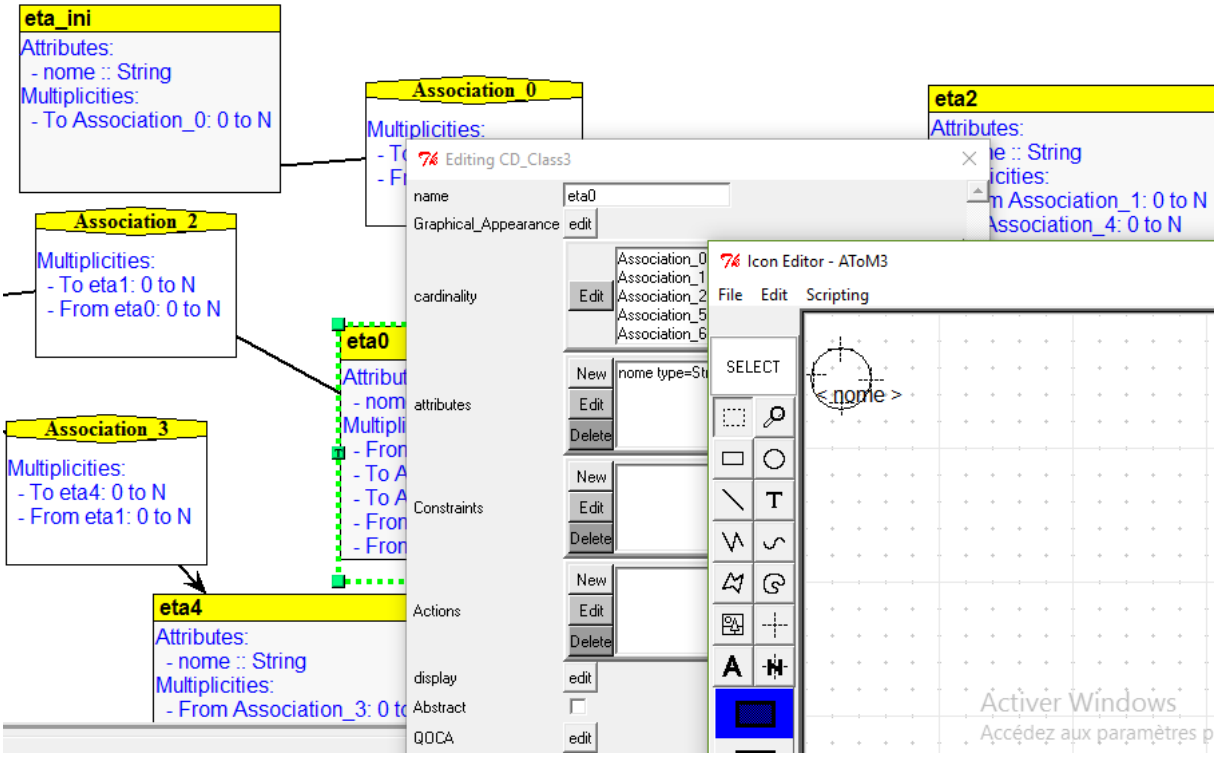

FIGURE 4.28 – Editeur graphique d'une action2

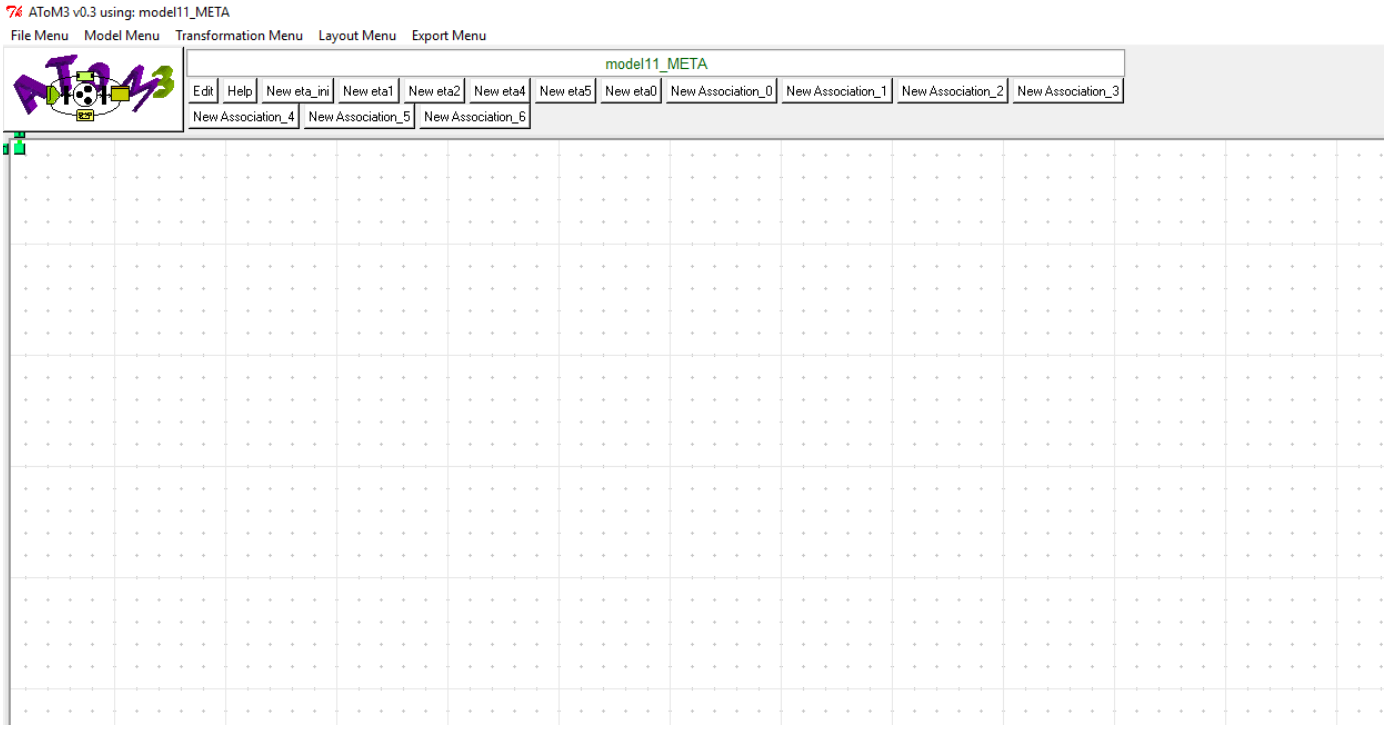

FIGURE 4.29 – Le méta modèle2 obtenu

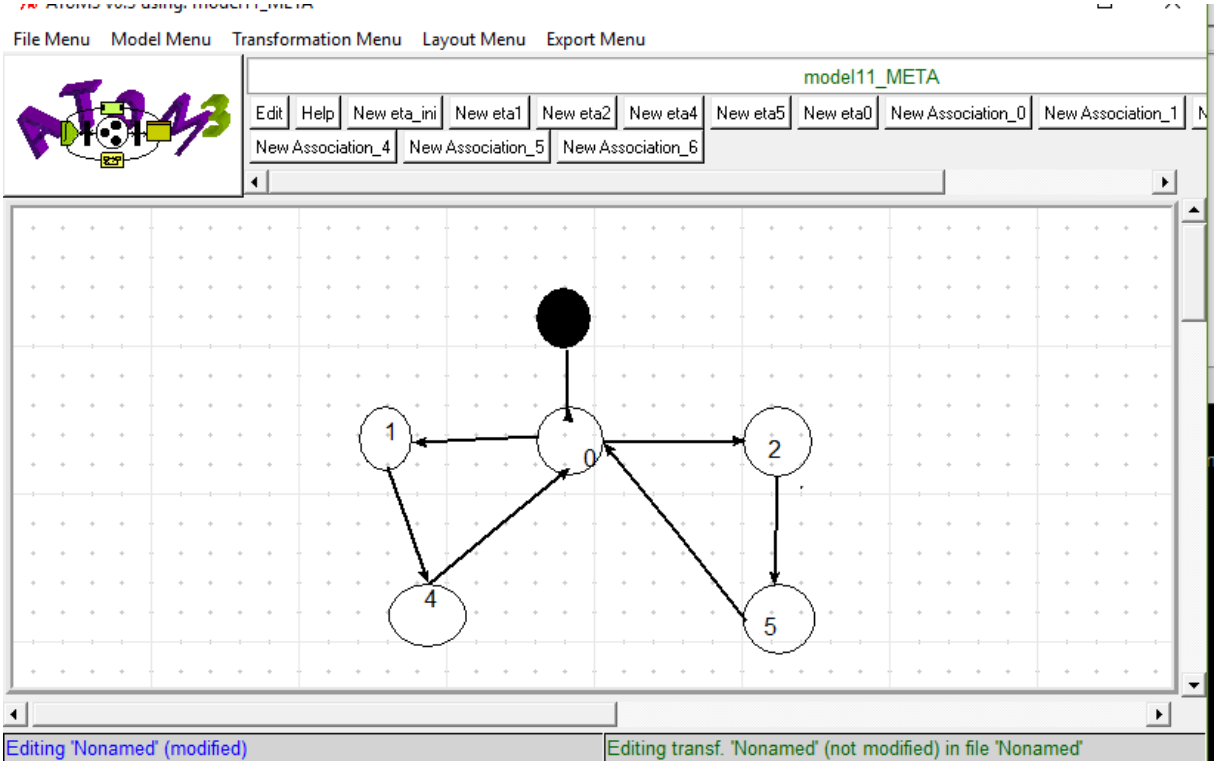

FIGURE4.30 -Réalisation des relations entres les états2.

Pour une autre Contraintes temporelles:

Contraintes temporelles:

- LRL, AV, et URL

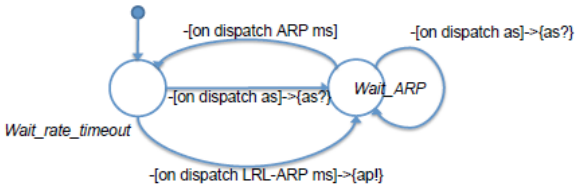

FIGURE 4.31 – Diagramme de Contraintes temporelles3

Le diagramme de classe équivalant est le suivant.

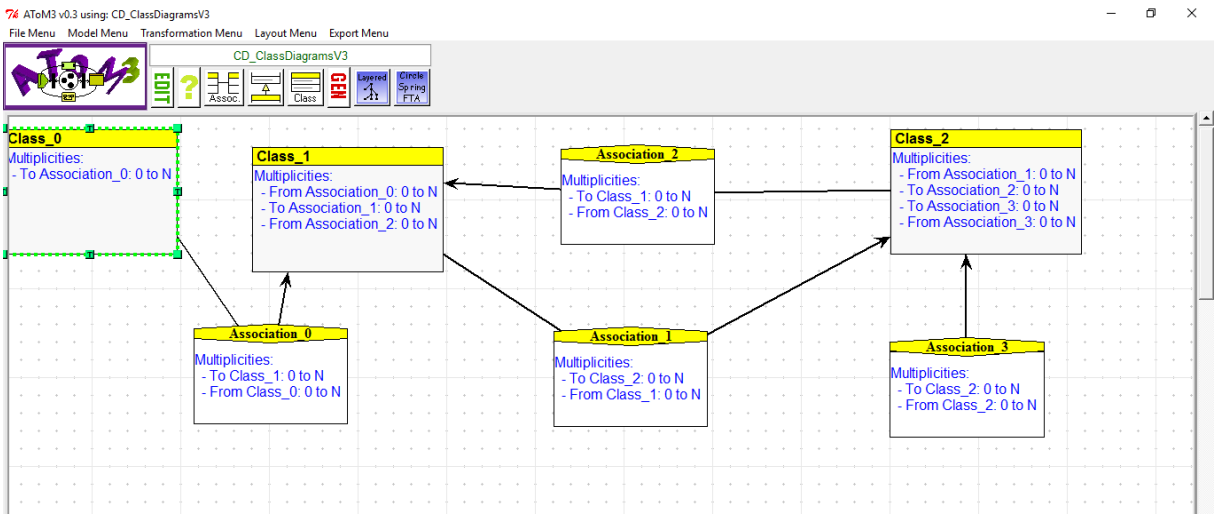

FIGURE 4.32 – Le diagramme de classe3.

Pour le méta modèle de ce modèle.

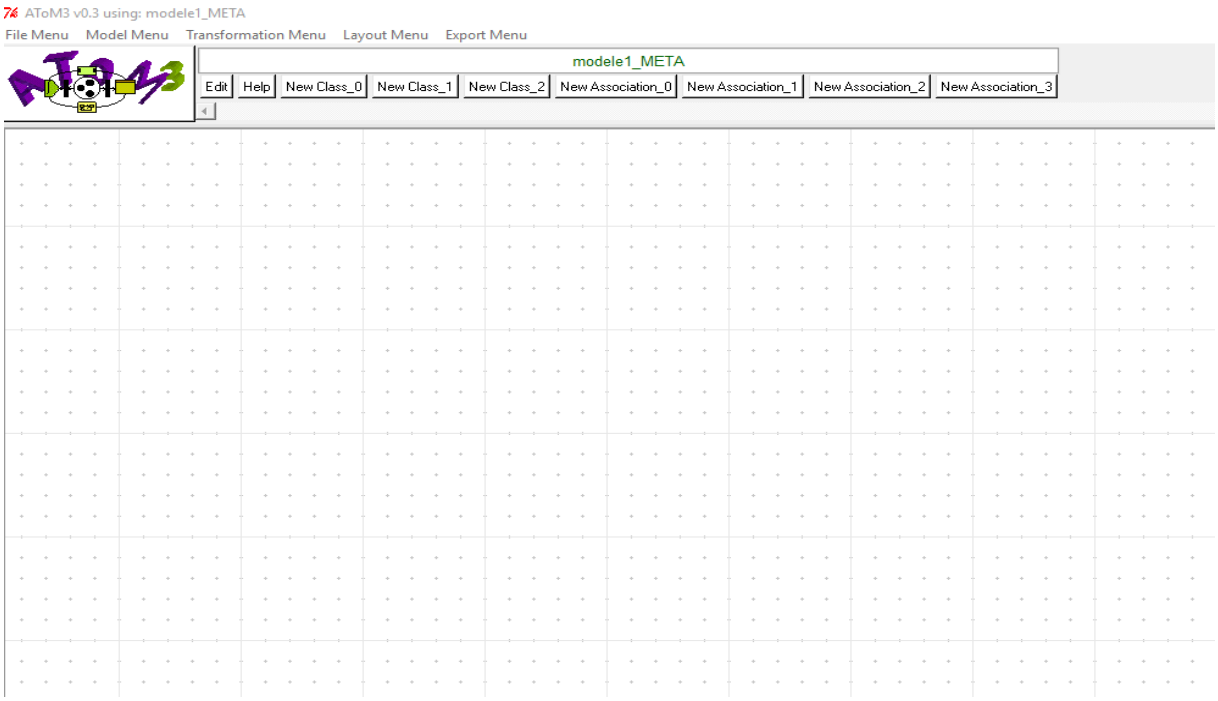

FIGURE 4.33 – Le méta model de modèle3

Puis redémarrer atom3 pour passer à une autre étape c'est les grammaires sur lesquelles on va parler dans les lignes suivantes car l'aspect de modélisation multi-formalismes de l'AToM3 est dû au fait qu'il permet de transformer un modèle dans un formalisme donné vers un autre formalisme en utilisant les grammaires de graphes.

Transformation de graphes :

Dans AToM3, un modèle chargé peut être transformé à un autre modèle équivalent, exprimé par un autre formalisme ou dans le même formalisme a l'aide du règles. un ensemble

de règles construire une grammaire de graphes. Elle est dotée d'une action initiale et une action finale. Elle a un nom et un ensemble de menus permettant sa manipulation, tel que la sauvegarde, l'exécution…etc. en peut voir tout ca dans la figure suivant :

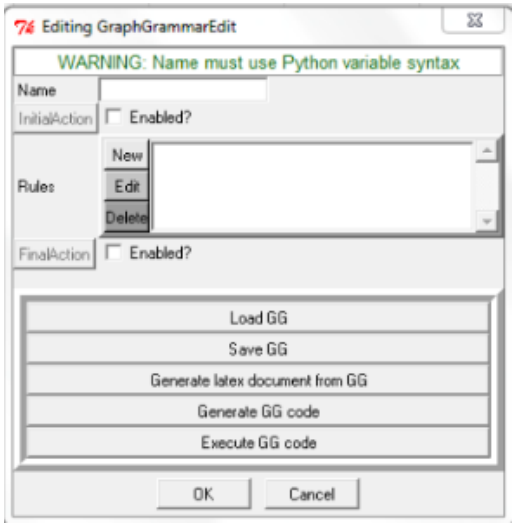

Figure 4.34 Editeur de grammaires d'AToM3

Chaque règle est constituée de:

- Un nom spécifique pour la règle.
- Une priorité indiquant l'ordre dans lequel la règle est appliquée.
- Une partie gauche (Left Hand Side : LHS) qui est un graphe.
- Une partie droite (Right Hand Side : RHS) qui peut être un graphe.
- Une condition (Un code en Python) qui doit être vérifiée avant que la règle soit appliquée.
- Une action (Un code en Python) qui doit être exécutée une fois que la règle est appliquée.

Les deux parties gauche et droite peuvent être des graphes illustrés par des formalismes différents, et pour faire la transformation il faut charger les méta-modèles qui concernent ces formalismes. L'exécution de règles peut être continue (aucune interaction d'utilisateur) ou étape par étape. Dans ce dernier cas, l'utilisateur doit intervenir après chaque exécution d'une règle pour passer à l'exécution de la règle suivante.

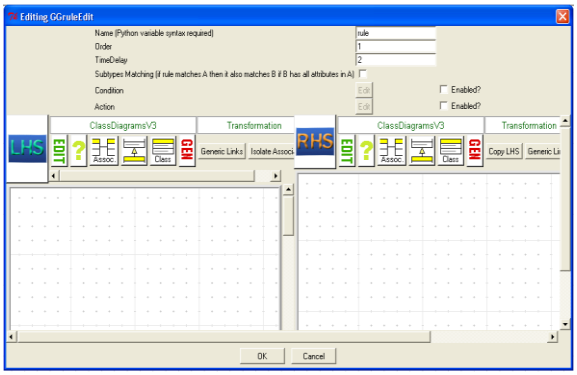

Figure 4.35 Editeur de règles d'AToM3.

On peut appliquer ca sur l'exemple précèdent Donc il faut ouvrer d'abord le méta modèle du modèle que nous avons déjà fait.

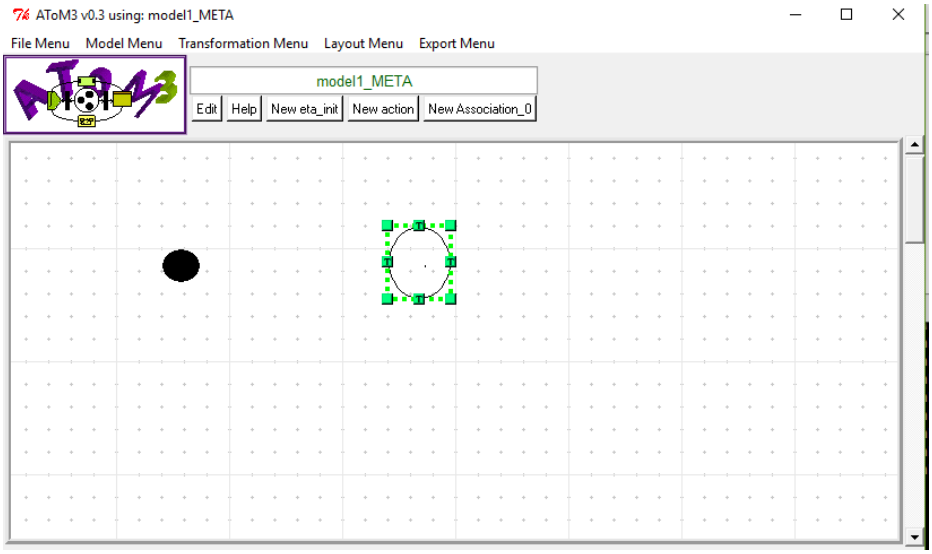

Figure 4.36 le méta modèle du modèle

Ensuit nous créons notre règle en choisissant transformation menu puis creat transformation ou bien en pressant 'T' dans le clavier directement.

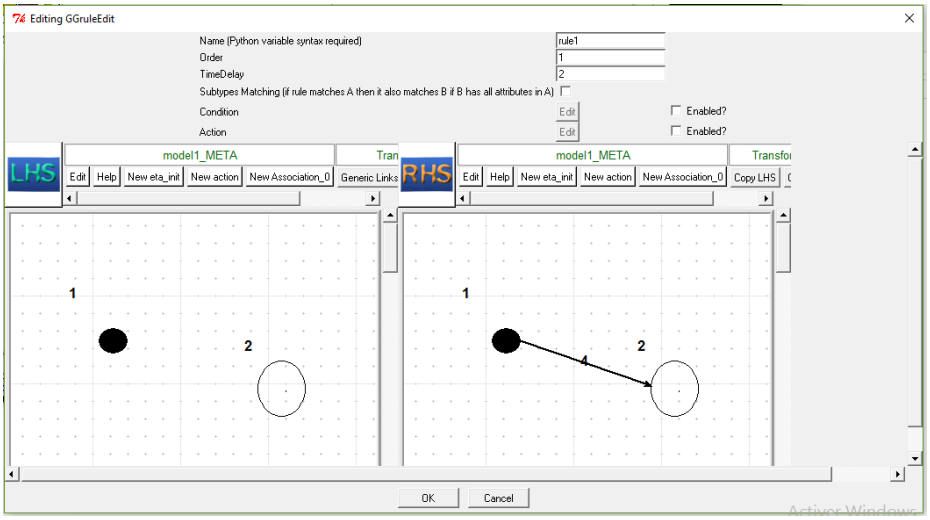

Figure 4.37 la création de la règle dans l'éditeur de règles d'AToM3.

Nous réalisons la règle en respectant l'ordre et les conditions après il faut sauvegarder et générer cette règle et finalement l'exécuter.

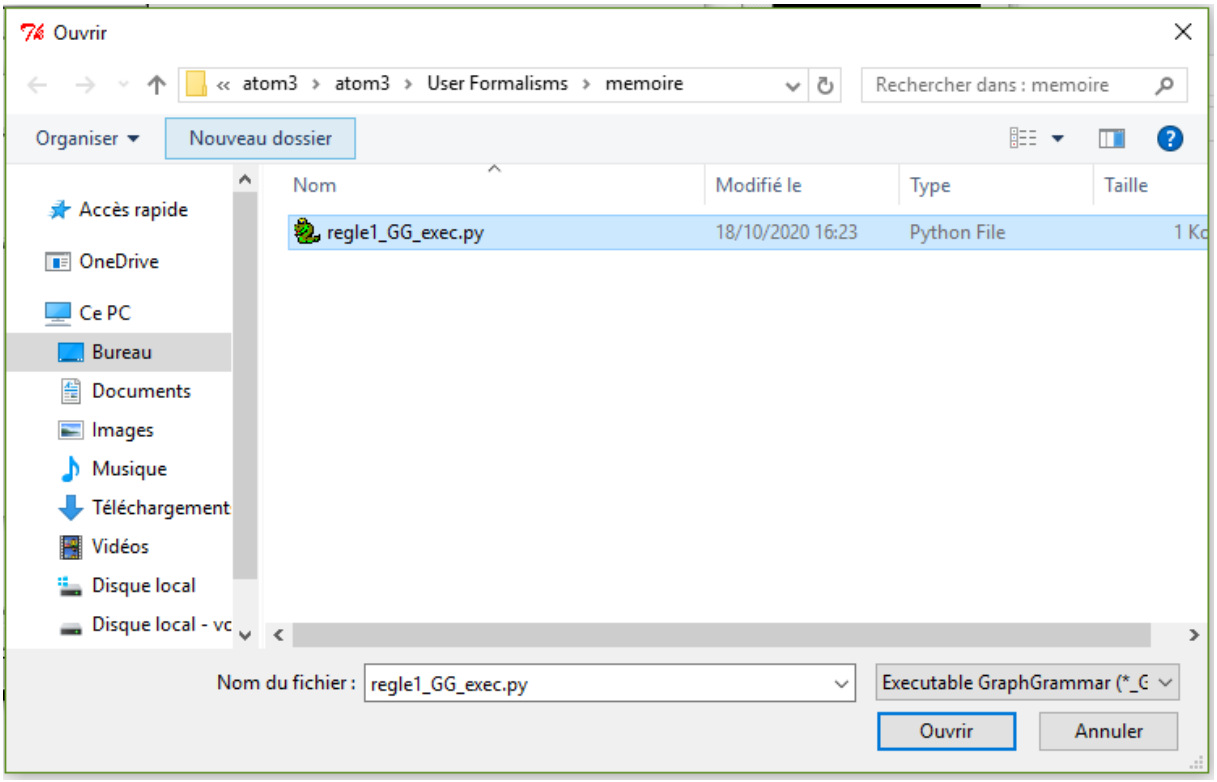

Figure 4.38 la sauvegarde de la règle.

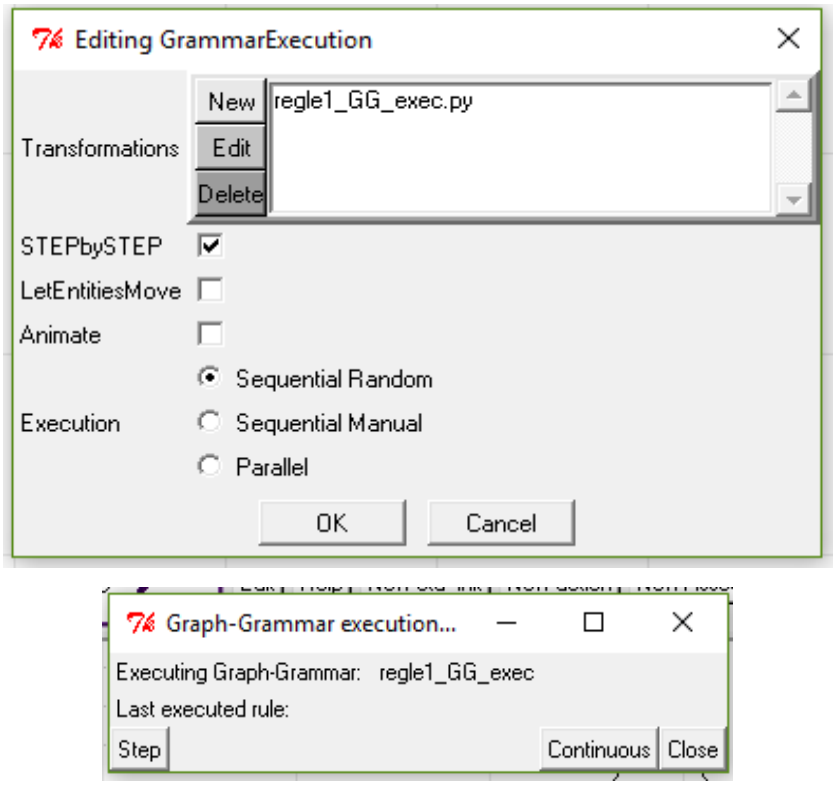

Figure 4.39 l'exécution de la règle

Donc nous obtenons le résultat suivant:

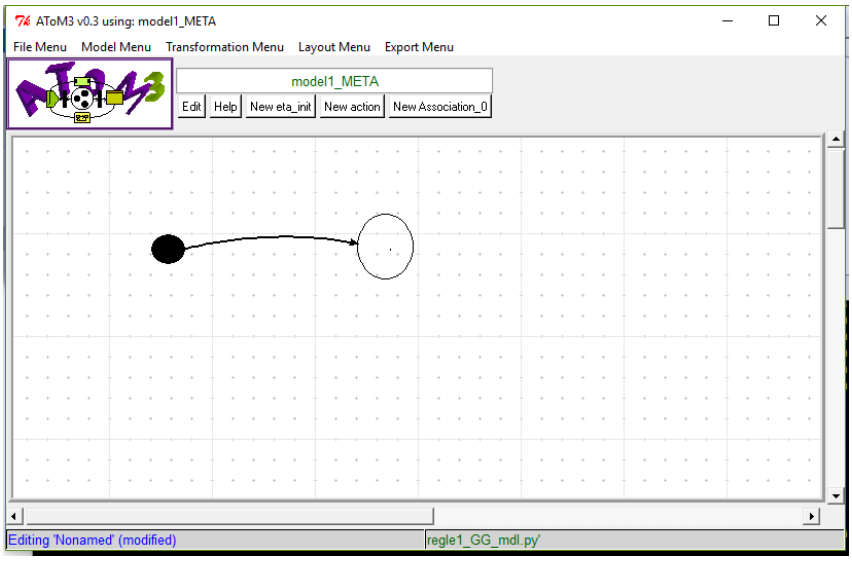

Figure 4.40 le résultat de l'exécution de la règle

Conclusion

Dans ce chapitre nous avons procédé à la mise en œuvre de notre étude de cas qui est le pacemaker avec notamment l'utilisation des outils Dia et Atom3.

L'objectif visé par ce projet de fin d'étude est d'étudier les systèmes cyber-physiques dans un but de modélisation.

Les systèmes cyber-physiques sont très complexes, ils sont composés de sous-systèmes interagissant entre eux pour avoir un tout cohérent. La compréhension et la mise en œuvre de la conception de ce type de système est une tâche très ardue.

Nous avons pris, comme étude de cas des systèmes cyber-physiques, le pacemaker qui est un dispositif électronique implanté dans le cœur d'un patient souffrant de bradycardie ou à l'inverse de tachycardie. Nous avons détaillé sa description, son cycle de vie et son comportement,

Nous avons opté pour AADL afin de faire la description du pacemaker et nous avons utilisé Dia et Atom3 dans la phase de mise en œuvre.

Comme perspectives à ce projet nous proposons de réaliser des transformations vers des modèles formels afin d'analyser et de tester les contraintes liées au fonctionnement du système.

Ce travail étant une œuvre humaine, ce n'est pas un modèle parfait, c'est pourquoi nous restons ouverts à toutes les critiques et nous sommes prêts à recevoir toutes les suggestions et remarques tendant à améliorer davantage ce travail, étant donné que tout travail informatique a été toujours l'œuvre d'une équipe.

[1] BNP PARIBAS ,«Les Etats-Unis misent gros sur la cyber-physique », Paris, L'Atelier, 3 septembre 2008 .

[2]Edward Lee, Systèmes Cyber physiques : Conception défis.

[3] Gilles Lasnier, Janette Cardoso, Claire Pagetti, Pierre Siron, « Environnement de coopération de simulation pour la conception de systèmes cyber-physiques », Rennes, Modélisation des systèmes réactifs (MSR), 2013 (consulté le 1er mai 2014)

[4] « Cyber-physical systems », The National Science Foundation, 4201 Wilson Boulevard, Arlington, Virginia 22230, USA, 30 septembre 2008 (consulté le 1er mai 2014)

[5] Claus Hilger, « La 4ème révolution industrielle est en marche », Tec news (consulté le 1er mai 2014)

[6] Wayne Wolf, The Good News and the Bad News (Embedded Computing Column), vol. 40, novembre 2007, 104 p. (DOI 10.1109/MC.2007.404), chap. 11

[7] « NSF Workshop On Cyber-Physical Systems » (consulté le 1er mai 2014)

[8] « Cyber Physical Systems for Europe CPS4EU »

[9] Olivier Cardin. Contribution à la conception, l'évaluation et l'implémentation de systèmes de production cyber-physiques. Automatique / Robotique. Université de nantes, 2016.

[10] Modélisation des systèmes temps-réel embarqués en utilisant AADL pour la génération automatique d'applications formellement vérifiées, Mohamed Yassin CHKOURI, 2010

[11] OMG Model-Driven Architecture Home Page, http ://www.omg.org/mda.

[12] Object managment group home page, http ://www.omg.org.

[13] SAE. Architecture Analysis & Design Language (standard SAE AS5506), September 2004, http://www.sae.org.

[14] Tewfik Ziadi, **« Langage et méthode pour une ingénierie des modèles fiable »**, 215p, Thèse de doctorat en informatique, Université de Renne1, 2006.

[15] Eveline Chantal Kabore, **« Contribution à l'automatisation d'un ons de processus de construction d'abstractions de communication par** 08. **transformations successives de modèles »**,222p, Thèse de doctorat en informatique, Université de Renne1, 2008.

[16] Sylvain Andre, **« MDA (Model Driven Architecture) principes et états de l'art »**, 25p, Mémoire d'ingénieur en informatique, C.N.A.M.Centre d'enseignement de Lyon, 2004.

[17] Object Management Group (OMG), **« MDA guide version1.0.1 »,** http://www.omg.org/docs/omg/03-06-01.pdf , juin 2003.

[18] Sébastien Revol, **« Profil UML pour TLM: contribution à la formalisation et à l'automatisation du flot de conception et vérification des systèmes-sur-puce »**,161p, Thèse de doctorat en Micro et Nano Electronique,Institut National Polytechnique de Grenoble, 2008.

[19] BOSTON SCIENTIFIC, PACEMAKER System Specification, janvier 2007].

[20] Embedded Systems: Analysis and Modeling with SysML, UML and AADL, Fabrice KORDON, Jérôme HUGUES, Agusti CANALS, Alain DOHET, WILEY 2013] .