République Algérienne Démocratique et Populaire
Ministère des études supérieures et de la recherche scientifique

Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

*Faculté des Sciences.*

***Mémoire de Master***
***En Informatique***
***Option : Génie Logiciel***

***Thème :***

# La mise en place de tests de spécification : le cas pratique d'un simulateur chirurgical.

***Présenté par :***

*BENKEBIL Hamza*

*BENHABIB Sidi Mohammed Abdelkader El Mehdi*

***Date de la soutenance :*** *24 / 06 / 2020*

***Membre du jury :***

| | | |
|---|---|---|
| *MALTI Djawida.* | *(Université de Tlemcen)* | **Président.** |
| *SELADJI Yasmine.* | *(Université de Tlemcen)* | **Encadrant.** |
| *MALTI Abed.* | *(Université de Tlemcen)* | **Encadrant.** |
| *MESSABIHI Mohamed.* | *(Université de Tlemcen)* | **Examinateur.** |

***Année universitaire***

***2019 – 2020***

# Dédicaces

***Louange à Dieu tout puissant***
***J'ai l'honneur de dédier ce modeste travail :***

*A celle qui m'a donné la vie, qui m'a soutenue dans les joies et les peines :*

*Ma mère ;*

*A celui qui m'a encouragé, éduqué et enseigné*

*Mon père ;*

*A mes sœurs Kawthar et Asma, et mon beau-frère Ramzi, mes neveux Yanis et Iyad ;*

*A tous les membres de ma famille*

*À cher Binôme et frère Hamza.*

*A l'ensemble de mes amis(es) et de mes camarades de promo chacun par son nom, en particulier Lakhdar, Youcef, et Islam*

***BENHABIB Sidi Mohammed.***

# *Dédicaces*

***Louange à Dieu tout puissant***
***J'ai l'honneur de dédier ce modeste travail :***

*A celle qui m'a donné la vie, qui m'a soutenue dans les joies et les peines :*

*Ma mère ;*

*A celui qui m'a encouragé, éduqué et enseigné*

*Mon père ;*

*A mes frères Abderrahman et Ghouti ;*

*A tous les membres de ma famille*

*À cher Binôme et frère Sidi mohammed.*

*A l'ensemble de mes amis(es) et de mes camarades de promo chacun par son nom, en particulier Lakhdar, Youcef, et Islam*

***BENKEBIL Hamza.***

# *Remerciements*

Nous tenons tout d'abord à remercier Dieu le tout puissant qui nous a donné la force et la patience d'accomplir ce modeste travail.

En second lieu, nous tenons à remercier nos encadrants Mme SELADJI Yasmine et Mr MALTI Abed pour leurs conseils, leur aide et leur patience pendant l'élaboration du mémoire. Leur soutien nous a été très précieux ce qui nous a permis de mener notre travail à bonne eussions.

Nos vifs remerciements vont également à Mr MESSABIHI Mohamed et Mme MALTI Djawida pour l'intérêt qu'ils ont porté à notre mémoire en acceptant de l'examiner et de l'enrichir par leurs propositions.

On aimerait aussi remercier nos parents les êtres les plus chers à nos cœurs, pour nous avoir prodigués conseils et encouragements, qui nous ont aidés à suivre nos études dans les meilleures conditions et qui nous ont toujours soutenus et encouragés sans limite.

Aussi, nous devons tant à nos familles et amis pour leur soutiens et conviction indéfectible, sans lequel nous ne pouvions accomplir tant de choses.

*Merci*

# ملخص

إن استخدام الواقع الافتراضي في بيئة تعليمية يحسن التعلم ويقلل من معدل الأخطاء الطبية للجراح ، ولكن هذه الأنواع من الأنظمة يمكن أن تؤدي إلى مخاطر جديدة يمكن أن تسبب إصابات خطيرة أو حتى وفاة المرضى إذا التعلم لا يتوافق مع الواقع. لذلك تعتبر سلامة النظام على أنها خاصية حاسمة للنظام .ومن هنا جاءت أهمية دراستنا التي تتمثل في النموذج تحديد مواصفات محاكي الجراحة لإجراء عمليات التحقق باستخدام تقنية فحص

# Résumé

L'utilisation de la réalité virtuelle dans un cadre pédagogique améliore l'apprentissage et réduit le taux des erreurs médicales du chirurgien.

Ce genre de systèmes peut introduire de nouveaux risques qui peuvent provoquer des blessures graves ou même la mort des patients si l'apprentissage n'est pas conforme à la réalité. Ainsi, la sureté a été perçue comme une propriété cruciale du système.

L’intérêt de notre étude consiste à mettre en place les spécifications d’un simulateur de chirurgie pour effectuer les vérifications en utilisant la technique du model-checking.

# Abstract

The use of virtual reality in an educational setting improves learning and reduces the rate of medical errors for the surgeon, but these types of systems can introduce new risks which can cause serious injuries or even death of patients if the learning does not conform to reality. Therefore, safety was seen as a crucial property of the system. Hence the interest of our study which consists in setting up the specifications of a surgery simulator to perform verifications using the model-checking technique.

# Table des matières

# Figures

# Tableaux

# Terminologies

| Termes | Définitions |
|---|---|
| Sommets : | Points ou de mailles interconnectés qui composent l'organe. |
| Déformation : | Déplacement des sommets qui composent l'organe Durant la simulation. |
| Dissection : | Changement de topologie de l'organe / séparation des sommets qui composent l'organe |
| Agrippement : | Les mailles qui composent l'organe qui sont touchés par l'outil deviennent lier |
| Endommagement : | Déformation qui a des répercutions fatales sur la vitalité de l'organe. |
| Position virtuelle de l'organe : | Position de l'organe dans la simulation dans l'espace (x,y,z). |
| Position virtuelle de l'outil : | Position de la manette dans la simulation (x,y,z). |
| Position virtuelle du sommet : | Position d'un point qui compose l'organe dans l'espace (x,y,z). |
| Position de la manette : | La position de la manette dans l'espace (x,y,z). |
| Fd : Force de déformation : | Valeur de la force requise pour calculer une déformation. |
| Fdi : Force de dissection : | Valeur de la force requise pour produire une dissection. |
| Fa : Force d'agrippement : | Valeur de force nécessaire pour agripper. |
| Fe : Force d'endommagement : | Valeur de force requise pour endommager. |

# Introduction Générale

Le 21eme siècle est le siècle du numérique et par conséquent de l'informatique, parmi ses domaines, nous trouvons la réalité virtuelle, qui se base sur l'utilisation de systèmes embarqués afin d'assurer l'immersion dans un environnement virtuel, l'interaction avec cet environnement en temps réel, créent des contraintes temporelles qui qualifierait ces derniers de systèmes temps réel.

La réalité virtuelle est utilisée dans divers domaines, entre autres, le domaine médical où elle est utilisée pour effectuer des thérapies, diagnostiques, et, dans un cadre pédagogique ou elle est utilisée pour apprendre aux chirurgiens comment effectuer diverses opérations chirurgicales. Ce qui exige une reproduction des opérations dans un environnement régi par les mêmes lois de physique que celle dans le monde réel d'où la nécessite de vérifier la sureté du système.

Dans le cadre de notre mémoire, qui consiste en « la mise en place de tests de spécifications », nous avons choisi de traiter le cas d'un système de simulation chirurgical « VRAnat » qui est en cours de développement par une équipe de doctorants encadrés par Monsieur MALTI Abed, professeur et enseignement au sein de la faculté de technologie de Chetouane Tlemcen, en vus du besoin d'assurer la sureté des interactions du médecin avec le simulateur.

# Problématique

Comme tout système embarqué opérant en temps réel, le simulateur chirurgical « VRAnat » est confronté à des contraintes temporelles, dû aux latences entre l'action de l'utilisateur et les retours sensoriels durant la simulation. Dans ce système la sûreté constitue un facteur essentiel pour assurer la fiabilité des connaissances acquises à travers la simulation qui dans le cas contraire provoquerait des résultats catastrophiques durant les opérations, L'intérêt de notre étude consiste en la mise en place de tests de spécifications, qui constitue une étape primordiale pour vérifier que les interactions entre les composants du système qui reflètent un environnement naturel que ce soit à travers la perception visuelle, haptique, ou sensorielle du chirurgien.

# Objectifs

Afin de répondre aux problématiques liées à la vérification et la validation du système avec ses contraintes temporelles et les retours sensoriels, nous avons choisi l'approche des méthodes formelles pour mettre en place nos tests de spécifications. De ce fait notre étude vise à analyser les besoins du système afin de les vérifier et assurer sa sureté.

# Chapitre I Généralités sur la réalité virtuel et la vérification

# I.1 Introduction

Ce chapitre présente quelques notions générales sur le domaine de notre étude tell que la réalité virtuelle, la spécification des besoins, la vérification et la validation ainsi que des notions de base sur le modèle checking.

# I.2 La réalité virtuelle

Dans cette section, nous allons voire qu'est que la réalité virtuelle ; Ainsi que quelques domaines d'application de la réalité virtuelle tell que le domaine de la médecine.

## I.2.1 Définition

La réalité virtuelle (ou Virtual Reality en anglais) est une expression qui désigne les dispositifs permettant de simuler numériquement un environnement par la machine (ordinateur), selon les technologies employées, elle permet à l'utilisateur de ressentir un univers virtuel par le biais de ses différents sens : la vue le plus souvent, mais aussi le toucher, l'ouïe, l'odorat.[1]

La réalité virtuelle permet donc à une personne de vivre une expérience d'immersion et de mener une activité sens-motrice dans un monde artificiel.

Pour garantir une immersion totale, l'utilisateur se sert d'un casque de réalité virtuelle. Celui-ci utilise le principe d'affichage en 3D stéréoscopique pour placer le spectateur dans un monde virtuel généré par une machine.

## I.2.2 Domaines d'application

De nos jours la réalité virtuelle est devenue un domaine professionnel par excellence, on la trouve dans tous les domaines partant du monde des jeux vidéo jusqu'au monde professionnel et humain comme la médecine, voir ci-dessous quelques domaines d'utilisation de la réalité virtuel dans le monde professionnels.

### I.2.2.1 La réalité virtuelle dans l'immobilier :

La réalité virtuelle dans l'immobilier s'est peu à peu démocratisée, notamment par le biais de la visite virtuelle immersive et non-immersive. Celle-ci donne la possibilité de s'immerger dans un projet immobilier, améliorant alors grandement la capacité de projection des clients.

Les Visites à 360°, les interactions et le réalisme bluffant de la 3D viennent séduire de plus en plus de professionnels de l'immobilier. [2]

### I.2.2.2 La formation par la réalité virtuelle :

Autres usages dans l'industrie, les outils d'aide à la formation ou à la maintenance se multiplient. Ils permettent d'intégrer des process industriels dans un outil de réalité virtuelle et de les inculquer à la personne formée avant d'intervenir sur site. Manipulation, tutoriels scénarisés, les casques VR permettent de réaliser de véritables expériences immersives et interactives. L'intégralité du poste de travail est modélisée dans une 3D réaliste et immerge l'apprenant. De plus, des situations impossibles à simuler dans le réel peuvent être intégrer à la scénarisation : incidents, anomalies, risques, etc...

Côté matériel, les formations se font quasi exclusivement sur les casques VR dit « sédentaires » comme le HTC Vive ou l'Oculus Rift. Disponibles avec des manettes de contrôle, ils permettent d'interagir directement sur l'environnement virtuel de manière fidèle au réel. Le fait de pouvoir répéter autant que voulu la formation en VR est également un gain non négligeable dans la phase d'apprentissage. [1]

### I.2.2.3 La réalité virtuelle dans la médecine :

Comme cité dans [3] une étude menée par UCLA pour comparer l'entraînement classique et celui en réalité virtuelle des chirurgiens montre une amélioration des capacités plus que conséquente.

C'est une étude qui devrait faire du bruit quant à son potentiel pour améliorer l'entraînement en réalité virtuelle. Menée par David Geffen de l'école de médecine de l'université de Californie, elle est intitulée « *Essai aléatoire et contrôlé d'un outil de réalité virtuelle pour enseigner la technique chirurgicale du clouage intramédullaire d'une fracture de la tige tibiale* ».

L'étude a été réalisée sur un groupe limité de 20 élèves chirurgiens étudiants, ce qui amène aussi à la relativiser pour l'instant. 10 d'entre eux se sont entraînés avec Osso VR, un casque de réalité virtuelle et des contrôleurs. Les autres avec les méthodes classiques. Après leur entraînement ils ont réalisé une opération sur un os d'entraînement artificiel et leur

performance a été filmée puis évaluée par un chirurgien professionnel selon plusieurs critères : temps, mouvements, gestions des outils, connaissance des outils, gestion de l'opération et connaissances spécifiques de l'opération

Le bilan est sans appel. Les étudiants chirurgiens entraînés par la réalité virtuelle ont été évalués 130 % plus haut que les autres. Ils ont aussi géré l'opération avec 38 % d'efficacité supplémentaire et 20 % plus rapidement. Pour l'instant, les résultats sont bien sûrs à observer avec prudence au vu de l'échantillon observé. Cependant, pour Osso VR qui défend l'approche de son outil par rapport aux méthodes classiques, il s'agit d'un vrai succès. La réalité virtuelle en médecine qu'il s'agisse de l'apprentissage ou pour les professionnels a de beaux jours devant elle.[3]

## I.3 Analyse et spécification des besoins

Dans cette section, on va voir ce qu'est une spécification par rapport à un besoin, et quel est son rôle et impact dans notre étude.

### I.3.1 Définitions

Une spécification signifie la définition des caractéristiques essentielles (qualité, dimensions, etc.) que doit avoir une marchandise, une construction, un matériel, etc. [4]

En informatique : "la spécification ou analyse des besoins est un modèle d'un logiciel déterminé à travers l'analyse des spécifications qui est une étape en génie logiciel qui consiste à décrire ce que le logiciel doit faire. Plus généralement une spécification peut aussi représenter n'importe quel système dynamique comme un circuit électronique"[5]

### I.3.2 Rôle

L'analyse des besoins est une étape indispensable dans le développement informatique, c'est l'étape sur laquelle repose toute l'étude et implémentation qui suit [6], elle permet de :

- Identifier les différents acteurs dans le système.
- Identifier le rôle de chaque acteur et aspect fonctionnel dans le système.

- Structurer les besoins du client.
- Etablir le cahier de charge.
- Valider le produit.

## I.3.3 Type de spécifications

On distingue 3 types de spécifications [7] :

- Les spécifications informelles : sont écrites sous forme de texte en français par exemple.
- Les spécifications semi-formelles : sont un peu plus précises par rapport aux premières, on peut attacher les textes avec des diagrammes ...etc.
- Les spécifications formelles : sont définis par une syntaxe et une sémantique écrite en langage formel.

PS : les spécifications font parties du domaine du génie logiciel et des méthodes Formelles.

## I.3.4 Spécification des besoins

La spécification des besoins est une étape primordiale dans le cycle de vie du développement, du fait qu'elle consiste à étudier les besoins du client et qu'est-ce qu'il veut exactement.

Dans cette phase on détermine les fonctionnalités et les acteurs les plus pertinents, on détermine aussi quels sont les risques les plus critiques, et on identifie les cas d'utilisations initiaux.[8]

### I.3.4.1 Besoins fonctionnels et besoins non-fonctionnels

Après avoir collecter les informations et détecter les besoins, il est important de les classer selon leurs types, besoins fonctionnels et non-fonctionnels.[9]

Besoin fonctionnel : cela veut dire un besoin dont le système a besoin, et qu'il ne peut être fonctionnel que lorsque que ce besoin est inclus. Par exemple : dans un MP3 le système doit jouer de la musique.

Besoin non-fonctionnel : contrairement au premier type, ce besoin est moins important, mais non négligeable, il s'agit des mesures de la performance, de la convivialité ...etc., et que

le système doit la mettre en valeur. Par exemple : Dans un MP3 le système doit jouer une musique rapidement, une seconde après le click.

### I.3.4.2 Déclaration des besoins fonctionnels

La déclaration des besoins fonctionnels se fait avec un diagramme de “use case”, c’est un diagramme de UML très pratique et très puissant, il permet de mettre en avant les besoins fonctionnels ainsi que les acteurs impliqués dans un système [10]. La Figure 1 Ci-dessous présente un exemple du cas d’utilisation d’un radio-réveil. L’utilisateur peut être réveillé à l’heur et peut écouter la radio grâce à l’émetteur radio, il peut aussi avoir l’heure.

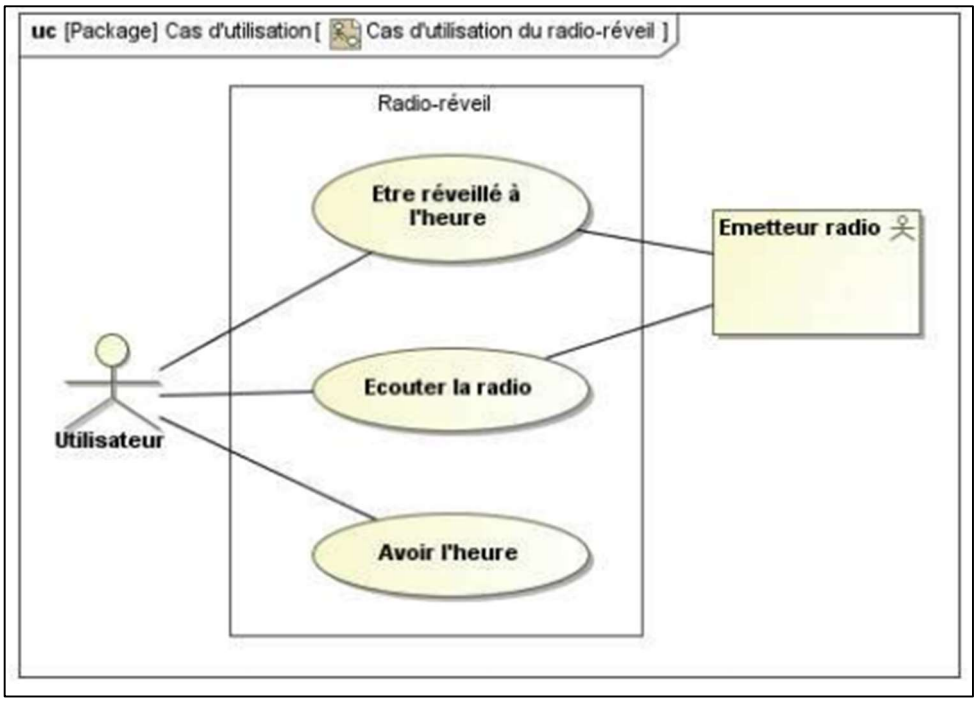


*Figure 1. Diagramme de cas d'utilisation du système de radio réveil*

# I.4 Validation et vérification

## I.4.1 Définition

La méthode qui permet de déterminer si un produit satisfait ou non ses exigences et ses spécifications La vérification est, autrement dit c'est la réponse à la question : construisons-nous le système correctement ?

La validation est l'évaluation d'un produit à livrer s'il répond aux besoins opérationnels du client dans un environnement réaliste, en outre c'est la réponse à la question : construisons-nous le bon système ? [11],[28]

## I.4.2 Les erreurs/défauts/défaillances

Un dysfonctionnement désigne un comportement inattendu du système qui serait dû à des bugs ou anomalies, ainsi ces anomalies sont classées selon leurs causes, et conséquences en trois types comme démontré dans [12] :

Les erreurs : Un dysfonctionnement est dit une erreur si le system fait autre que ce qui lui est demande de faire, que ce soit dû à une mauvaise compréhension des besoins ou s'ils sont mal exprimés dans le cahier de charge.

Les défauts : Un défaut se produit lorsqu'une exigence est oubliée ou mal spécifiée, même si un défaut n'est pas forcement critique et ne cause pas toujours une défaillance.

Les défaillances : sont des dysfonctionnements caractérisés par des résultats erronés et des services non rendus. Les défaillances sont souvent critiques car elles apparaissent en production, et peuvent engendrer des pannes.

### I.4.3 Aperçu

Dans la phase de développement, les procédures de vérification incluent des tests spéciaux pour modéliser ou simuler une partie ou la totalité d'un produit, service ou système, puis vérifier ou analyser les résultats de la modélisation. Au stade post-développement, les procédures de vérification impliquent de répéter régulièrement des tests spécialement conçus pour garantir que les produits, services ou systèmes continuent de répondre aux exigences de conception, spécifications et réglementations initiales au fil du temps. Il s'agit d'un processus utilisé pour évaluer si un produit, un service ou un système est conforme aux réglementations, spécifications ou conditions imposées au début de la phase de développement. La vérification peut être effectuée pendant le développement, l'expansion ou la production. Il s'agit généralement d'un processus interne.

Parfois, quelqu'un dit que la vérification peut être exprimée par la demande "Êtes-vous en train de construire la bonne chose ?". Et cochez "Est-il construit correctement ?". "Construire la bonne chose" fait référence aux besoins de l'utilisateur et "construire la bonne chose" vérifie si la spécification a été correctement mise en œuvre par le système. Dans certains cas, il est nécessaire d'avoir des exigences écrites pour les documents écrits et les procédures ou accords formels pour déterminer la conformité.

Le produit est susceptible de réussir après vérification, mais il est susceptible d'échouer après vérification. Par exemple, cela peut se produire lorsqu'un produit est fabriqué selon des spécifications mais que les spécifications elles-mêmes ne peuvent pas répondre aux besoins des utilisateurs.[13]

## I.4.4 Types des vérifications et de validation

On constate qu'il y'a deux types de V&V [14] :

**Vérification et Validation statique** :

Ce type permet d'analyser et testé un système sans devoir à l'exécuter, afin de découvrir ses problèmes.

**Vérification et Validation dynamique :**

Ce type permet d'analyser et testé un système avec exécution.

## I.4.5 Catégories des vérifications et validations

Revue du code : c'est la plus simple méthode de vérifier et détecter les erreurs d'un code source... mais ce n'est pas évident d'en tester un nombre de lignes assez important !

Tests unitaires : dans la vérification un test unitaire désigne une détection d'erreurs par unité, une exécution modulaire afin de valider si chaque partie répond correctement aux fonctionnalités définit préalablement.

Tests d'intégrations : une fois que les tests unitaires sont validés, on vérifie dans ce test si les unités communiquent correctement entre eux.

Tests fonctionnels de bout en bout : C'est l'étape qui suit intégration des modules, et ça permet au testeur de se mettre à la place de l'utilisateur et vérifie le système de bout au bout afin de valider si le comportement de l'application se comporte comme prévue.

Tests de charge : A travers ces tests les testeurs permettent de s'assurer que l'application va être capable de tenir la charge qu'on lui impose lors de l'exécution dans un environnement réel.

Tests d'exploitabilité : Dès que les tests précédents soient terminés le produit est prêt à être déployé en production, dès que le test d'exploitabilité sera validé, il a pour objectif de confirmer que l'application fonctionnera comme attendu sur l'environnement de production, ce teste se fait conjointement avec la DCI (Direction des Services Informatiques) [15]

## I.4.6 Logiques temporelles et Le Model-checking

### I.4.6.1 Définition

Le model-checking est une approche automatisée permettant de verrier qu'un modèle de système est conforme à ses spéciations. Le comportement du système est formellement modélisé, via des automates, réseaux de Petri, algèbres de processus, et les spéciations, exprimant les propriétés attendues du système, sont formellement exprimées par exemple via des formules de logiques temporelles. En pratique, les propriétés du système sont souvent classées en deux grandes catégories informelles. Les propriétés de sûreté énoncent qu'une situation particulière ne peut être atteinte. Les propriétés de vivacité énoncent quelque chose de mauvais (ou de bon) qui finira par se produire.[16]

Le model-checking permet de vérifier si une propriété est vérifiée dans l'automate qui représente le comportement de notre système. Cette vérification est effectuée en vérifiant si la formule est satisfaite dans l'automate.

La figure 2 montre un exemple d'un automate. L'automate montré à 4 états 1,2,3 et 4. Les transitions sont représentées avec des flèches. On peut par exemple aller de l'état 1 vers l'état 3 avec une transition B.

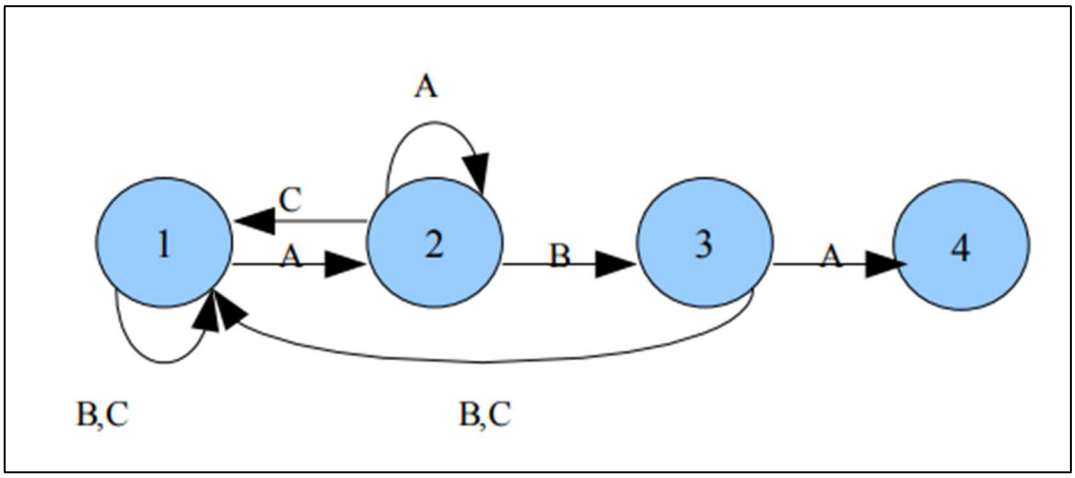


*Figure 2. Exemple d'un automate base sur le model checking*

L'utilisation du modèle model-checking suit un cycle de vie qu'on diviser en trois étapes qui sont répétées plusieurs fois jusqu'à ce que toutes les propriétés soient vérifiées par le système :

- La modélisation formelle du comportement sous forme d'automate comme dans la Figure 2

- L’expression formelle des propriétés attendu du système (Besoins)
- La détection de contre-exemple en cas de non-respect d’une propriété du système, afin de pouvoir l’analyser et de pouvoir produire une correction.

### I.4.6.2 Objectifs et Intérêt

Ce type de modèles sert à modéliser et vérifier des systèmes temps réel, réactifs, et critique d’une manière formelle en utilisant un langage formel (logique temporelle) et des propriétés, la vérification et la validation par ce genre de modèle est précise.[16]

### I.4.6.3 Définition formelle de l’automate (Structure de Kripke)

Un automate est habituellement représenté par un quintuple A =<Q,E,T,q,l> où :

Q : est un ensemble fini d'états,

e : est l'ensemble fini des étiquettes de transitions,

t : est l'ensemble des états des transitions,

q : est l'état initial de l'automate,

l : est l'application qui associe à tout état de Q l'ensemble des propriétés élémentaires

Vérifiées dans cet état.[17]

### I.4.6.4 Logique temporelle

Les logiques temporelles permettent d’exprimer des propriétés sur les états/transitions passés, présents ou futurs qu’un système peut/doit atteindre. A cet effet, elles offrent des opérateurs temporels spécifiques, des modalités, tels que finalement ou jamais, qui permettent de décrire des ordres entre les évènements/états sans pour autant introduire le temps explicitement. Pour ces logiques, le temps est une donnée abstraite qui n’est pas quantifier par opposition aux logiques temporelles temporisées qui permettent de manipuler explicitement le temps.[17]

Toute proposition logique temporel est composée de deux éléments :

AG Φ tous les chemins satisfont toujours Φ

AF Φ tous les chemins satisfont un jour Φ

AX Φ tous les premiers suffixes des chemins satisfont Φ

A (ΦU Φ’) tous les chemins satisfont Φ jusqu’à satisfaire Φ’

EG Φ il existe un chemin satisfont toujours Φ

EF Φ il existe un chemin satisfont un jour Φ

EX Φ il existe un premier suffixe des chemins satisfont Φ

E (ΦU Φ') il existe un chemin satisfont Φ jusqu'à satisfaire Φ'

- **Syntaxe**

CTL formules sont définit [20] par :

∀p ∈AP, p ∈CTL

p,q ∈CTL où ¬p,p∨q,p∧q ∈CTL

p,q ∈CTL où EXp,AXp,EGp,AGp,EFp,AFp,E[pUq],A[pUq] ∈CTL

- **Arbre d'exécution**

Comme signifie son nom, cet arbre représente toutes les exécutions possibles pour un système donné, les nœuds et les feuilles sont les états atteints, les flèches représentent les transitions.[20]

Voici quelques exemples des arbres

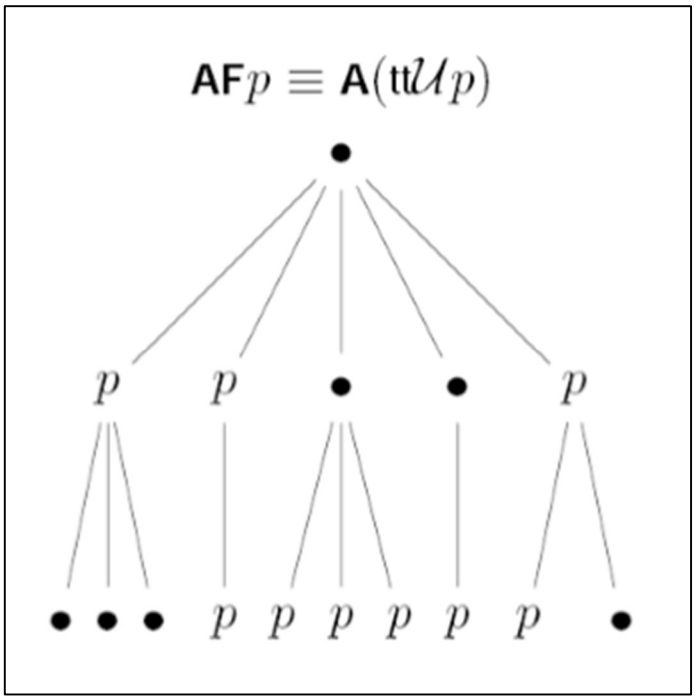


*Figure 3. Arbre d'exécution dans la logique temporelle simulant une exécution AFp*

Dans la Figure 2  Nous simulons l'exécution de la proposition temporelle AFp, qui signifie que tous les chemins satisfont toujours p, où quel que soit le chemin pris on atteint p .

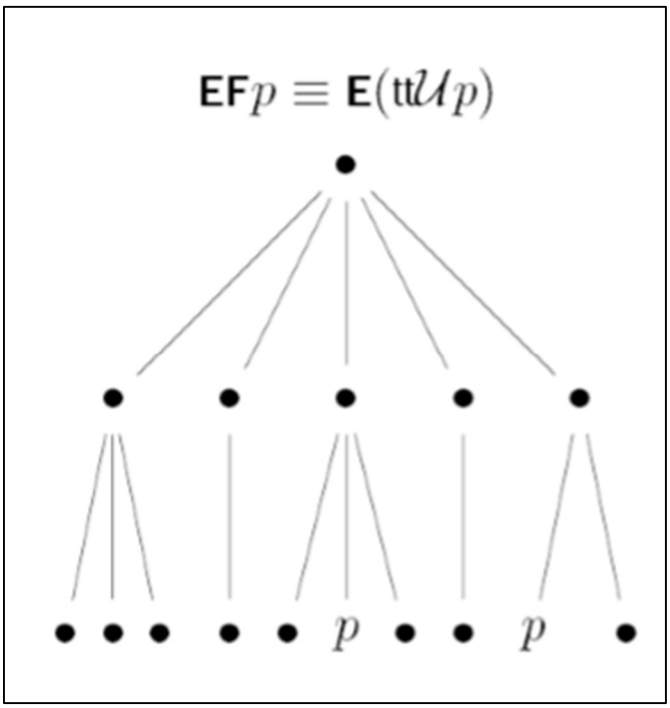


*Figure 4. Arbre d'exécution dans la logique temporelle simulant la proposition EFp*

Si on reprend la même logique avec la proposition EFp comme dans la Figure 4, qui signifie qu'il existe un chemin (en moins) dans le futur satisfait p, et on remarque aussi que lorsqu'on descend vers le bas de l'arbre on atteint en moins une fois p.

### I.4.6.5 Propriétés

La vérification par le modèle checking demande beaucoup d'acquis en théories et la mise en œuvre n'est pas un truc évident.[17]

Nous allons voir quelques types de propriété que les systèmes devront vérifier.

- **Propriétés d'atteignabilité**

On appelle une propriété d'atteignabilité une propriété qui vérifie la phrase suivante :

Dire qu'une certaine situation peut être atteinte.

Son rôle est très important parce qu'il spécifie une chose simple mais essentielle.

Si on prend l'exemple de l'ascenseur, nous pouvons vouloir spécifier que nous pouvons entrer une zone critique (remplacer le plancher). Mais en général, il est intéressant de considérer

l'inversion de ces attributs, Nous supposons que le système est dans un ou plusieurs "états crash", nous espérons Vérifiez que le système n'entrera jamais dans cet état.

Pour spécifier toutes les familles de propriétés on doit utiliser la logique temporelle.

Ici donc, on veut savoir si un état peut être atteint ou pas, pour le faire on utilise le connecteur logique EF Φ où :

Φ : une formule propositionnelle c'est à dire une formule qui n'a pas de connecteurs temporels.

Le connecteur EF veut dire qu'il existe un chemin où Φ sera vérifier.

- **Propriétés de sureté**

On appelle une propriété de sureté une propriété qui vérifie la phrase suivante :

Sous certaines conditions, quelque chose ne se produit jamais.

Si on reprend le cas de l'ascenseur, cela veut dire que l'ascenseur ne peut pas monter lorsqu'il est déjà en haut, dans la plupart des cas la négation d'une propriété de sureté résulte une propriété d'atteignabilité.

Si on applique le principe de la logique temporelle sur la propriété on aura les connecteurs AG¬Φ cela reprend celle d'une propriété d'atteignabilité.

- **Propriétés de vivacité**

On appelle une propriété de vivacité une propriété qui vérifie la phrase suivante :

Sous certaines conditions, quelque chose finira par avoir lieu

Toujours avec l'exemple de l'ascenseur cette proposition implique un exemple simple : si on appuie sur le bouton, alors la cabine arrivera un jour. On peut différencier maintenant entre la propriété de vivacité de celle d'atteignabilité, celle-ci est plus conteignante que l'autre. Et si on veut représenter l'exemple en haut avec la logique temporelle ça sera :

AG(Bouton=>AF AscenArrive) : AG : C'est à dire que tous les chemins vont satisfaire la phase qu'on a.

- **Propriétés d'équité**

On appelle une propriété d'équité une propriété qui vérifie la phrase suivante :

Sous certaines condition quelque chose aura lieu (ou pas) une infinité de fois.

On peut mettre un exemple d'un passage à niveau on l'on exprime le fait que la barrière se lèvera une infinité de fois. Sa représentation est A F ∞ P => AG AFP.

Vérification

La vérification se fait en explorant de manière exhaustive l'espace d'état, en déterminant si certaines spécifications sont vraies ou non.

L'idée ici est d'avoir un automate équivalent au modèle et un équivalent à la négation de la propriété. L'intersection de deux automatismes non déterministes tentent de faire échouer la propriété.

S |= ψ Si seulement si L(AS) ⊆L(Aψ)

Si seulement si L(AS) ∩L(A¬ψ) = ∅

### I.4.6.6 Exemple d'application

Prenons maintenant un exemple [20] concret pour résumer tous ce qu'on est en train d'expliquer.

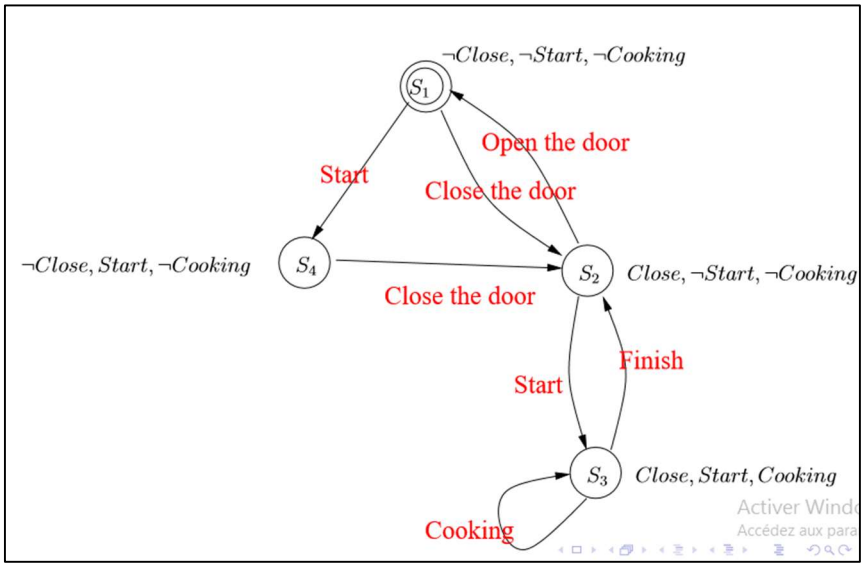


*Figure 5. Automate modélisant le comportement d'un micro-onde*

La figure ci-dessus représente un automate de graphe d'un micro-onde

Une structure de Kripke d'un micro-onde peut se représenter avec :

M = (S, s1, R, L)

S = {s1, s2, s3, s4 } s1 état initiale

R = {(s1,s2),(s2,s1),(s1,s4),(s4,s2),(s2,s3),(s3,s2), (s3,s3)}

L(s1) = {¬close,¬start,¬cooking},

L(s2) = {close,¬start,¬cooking},

L(s3) = {close,start,cooking},

L(s4) = {¬close,start,¬cooking}

On veut vérifier cette proposition

AG(start ⇒AFcooking)

AG veut dire que tous les chemins satisfont toujours start ⇒AFcooking , on peut la traduire par : quel que soit la situation lorsqu'on appuie sur le bouton Start, auras cooking dans le futur.

Vérifions cette proposition :

On doit réécrire sa négation ¬EF(start ∧EG¬cooking)

Calculons récursivement l'ensemble des états qui satisfont la formule en commençant par les formules atomiques :

S(start) = {s3,s4}

S(¬cooking) = {s1,s2,s4}

S(EG¬cooking) = {s1,s2,s4}

S(start ∧EG¬cooking) = {s4}

S(EF(start ∧EG¬cooking)) = {s1,s2,s3,s4}

S(¬EF(start ∧EG¬cooking)) = {}

PS ; on cherche toujours quels sont les états qui satisferont une propriété sur le graphe.

Puisque l'état initial s1 n'est pas contenu dans cet ensemble, nous concluons que la formule n'est pas satisfaite.

#### I.4.6.7 Outils

Comme on a fait cette vérification manuellement avec le graphe, cette vérification peut être faite automatiquement grâce à des outils, qui nous offrent une représentation et une vérification rapide et optimale à nos propriétés selon le contexte de notre système.

On trouve Kronos pour les automates temporelles, UPAAL pour les systèmes temps réels, CADP pour les systèmes distribués, SMV pour vérifier les automates temporelles à états fini tell que les conceptions du hardware, SPIN Pour LTL model checking.[20]

# I.5 Conclusion

Nous avons présenté dans ce chapitre une introduction sur les domaines de notre étude.

Ainsi nous avons passé en revue quelques concepts fondamentaux de la littérature sur la mise en place des spécifications, la vérification et la validation et la réalité virtuelle.

A cet effet, le lecteur qui souhaitera approfondir ces parties pourra se référer à ces deux livres [29],[30] qui traitent de façon détaillée la mise en place des spécifications et la vérification de logiciels critiques en utilisant les méthodes formelles basé sur la logique temporelle.

Cette étude nous a permis de poser des bases pour nos futurs travaux en situant progressivement le périmètre de ceux-ci. Dans le prochain chapitre nous traiterons des approches de vérification établies au paravent.

# Chapitre II Etat de l’art

# II.1 Introduction

Afin de réaliser notre étude, nous avons analysé des travaux similaires au notre, qui avait pour but d'effectuer des vérifications et des validations de différents systèmes critiques temps réel qui n'été pas forcement basé sur la réalité virtuelle, mais qui utilise une approche basée sur la vérification par les méthodes formelles et l'analyse des besoins.

Les tests de spécification représentent une phase très importante durant le développement des systèmes complexes, et encore pour des systèmes embarqués qui opèrent en temps réel.

C'est pour cette raison qu'il est impératif de se renseigner sur des systèmes similaires, de ce fait, nous avons choisis de décrire des systèmes qui ont servi de modèle à notre étude.

# II.2 La vérification des systèmes dans le domaine de l'industrie

L'article [20] présente la vérification des modèles comme méthode alternative pour estimer la latence et le parallélisme des systèmes interactifs asynchrones en temps réel (RIS). Plusieurs outils de vérification de modèle sont évalués par rapport aux exigences typiques dans le domaine RIS. Par conséquent, le langage formel de vérification de modèle Rebeca et son vérificateur de modèle RMC sont appliqués à la spécification des cas d'utilisation pour estimer la latence et le parallélisme de chaque use case. Il a également détecté un chemin d'exécution problématique non couvert par la nature stochastique des échantillons de profilage mesurés. Les résultats estimés du degré de parallélisation par vérification du modèle sont estimés avec une différence minimale de 9,3% et une différence maximale de -28,8%. Enfin, l'effort de vérification du modèle est comparé à l'effort d'implémentation et de profilage d'un RIS.

Un autre travail [21] explique que l'algorithme du modèle logique temporel est modifié pour représenter un graphe d'état en utilisant des diagrammes de décision binaires.
Cette technique est démontrée sur une conception en pipeline synchrone avec environ 5 * 10 / sup 20 / états. La logique utilisée pour spécifier les circuits est une logique temporelle propositionnelle du temps de branchement CTL ou Logic Tree Logic. L'algorithme de

vérification de modèle gère le CTL complet avec des contraintes d'équité. Par conséquent. il est possible de gérer un certain nombre de propriétés importantes de vivacité et d'équité. qui autrement ne serait pas exprimable en CTL.

Une nouvelle technique de vérification des systèmes interactifs expliqué dans un état de l'art [22]. Il présente d'abord l'utilisation d'une méthode formelle orientée modèle pour spécifier des systèmes interactifs, c'est-à-dire la méthode B. Ensuite, il propose des solutions formelles qui permettent de résoudre les difficultés inhérentes à la spécification des systèmes interactifs, comme l'accessibilité, l'observabilité ou la fiabilité. Nous affirmons que cette technique orientée modèle qui utilise des obligations de preuve peut être utilisée avec des techniques de vérification de modèle, où des preuves automatiques de propriétés peuvent être effectuées.

En outre, Les systèmes radar qui ont un large éventail d'applications dans les observations aériennes, navales ou de surface expliqué dans [23], nécessitent une précision et une fiabilité élevées, de ce fait, la vérification d'iinterface mémoire d'un système de radar.

Pour cette raison, des techniques de vérification formelles sont établi pour réduire les efforts et le temps, augmenter l'efficacité et concevoir un système de manière plus fiable. Ce système s'est concentré sur la technique de vérification basée sur un modèle. Les techniques basées sur des modèles sont basées sur des modèles décrivant le comportement du système d'une manière mathématique précise et sans ambiguïté. Après la modélisation précise du système, la vérification a lieu par simulation et vérification du modèle. Ils ont modélisé le modèle d'interface mémoire comme un automate temporisé utilisant l'environnement et les fonctionnalités UPPAAL.

## II.3 La vérification des systèmes dans le domaine de la médecine

Dans un système de contrôle élaboré pour les robotiques autonomes [24], les développeurs ont utilisé des méthodes formelles afin de vérifier ses propriétés, illustrant sont proposition dans un cadre d'une chirurgie en considérant l'exécution automatique d'une action simple telle que la perforation.

Pour prouver qu'une séquence de sous-tâches planifiées sur des données préopératoires peut réussir l'opération chirurgicale malgré les incertitudes du modèle, ils ont spécifié le problème en utilisant des automates hybrides, ils ont exprimé les exigences sous forme de

questions sur les propriétés du modèle de l'automate, ensuite ils ont utilisé l'outil Adriane pour vérifier les modèles des automates et détecter les erreurs qui affecte la sureté du système.

En outre, un article scientifique [25] montre les méthodes de la validation d'un robot dans les salles d'opération, cette technologie améliore la sureté et diminue le temps de récupération des patients et la fatigue du chirurgien, ce type de systèmes introduit de dangers potentiels qui peuvent entrainer des blessures graves ou même la mort de la personne. Comme la sureté a été perçu comme une propriété cruciale du système, dans cet article ils ont discuté de la technique de la validation formelle pour analyser les propriétés de sureté d'une étude de cas d'incision laser, en revanche, ils n'ont pas mentionné la méthode de la création du modèle du système et avec quel langage, le modèle du système prend la forme d'un automate hybride constitué d'une partie de commande discrète fonctionne dans un environnement non mentionné, les contraintes de sureté sont formalisées en tant que propriétés d'accessibilité du modèle d'automate hybride, tandis que la stratégie de vérification exploite les capacités de l'outil Ariane pour résoudre le problème de vérification et répondre aux questions connexes allant de la sécurité à l'efficience et l'efficacité.

## II.4 La vérification des systèmes dans le domaine de la réalité virtuelle

Le travail de recherche [26] explique la vérification et la modélisation du comportement opérationnel d'un système de tourisme en réalité virtuelle (VR) pour la conception architecturale au cours de la phase de définition des exigences système du cycle de vie du développement du système. Cette recherche vise à extraire les comportements les plus courants et initiaux des touristes, tels que la participation à des événements tels que la photographie, le parapente et la chasse. Ils ont évalué les propriétés opérationnelles complexes du processus de navigation, telles que la possibilité d'accidents lors de la participation à des événements. Cela soutiendra la conception et le développement de systèmes meilleurs et plus réalistes dans des conditions de fonctionnement réelles. En utilisant les réseaux de Petri et HierarchicalPetriNet Simulator De plus, ils ont généré un cadre de test basé sur les profils opérationnels pour améliorer la fiabilité de l'application VR.

# Chapitre III Contribution : Analyse et vérification du système

# III.1 Introduction

Dans ce chapitre, nous allons présenter dans un premier temps le principe général de notre approche et la base formelle qui sera utilisée par la suite dans notre étude. Ensuite, nous présenterons le processus de notre étude :

1) Analyse du système et extraction des spécifications.
2) Conception
3) La mise en place de l'automate de vérification
4) Vérification et validation.

# III.2 Analyse du système et extraction des spécifications

Dans cette section on analyse le système de la simulation chirurgical "VRnat" ; Afin d'en extraire les spécifications, on analyse ses modules, les entrés les sorties et les propriétés de chacun et on représente les spécifications sous forme de tableaux en dessous du module analysé.

La Figure 6 représente les composant du système de la simulation chirurgical "VRnat". Les objets interactifs sont représentés dans des rectangles avec des couleurs, les interactions entre les objets sont représentées avec des flèches et un texte explicatif de la nature de l'interaction. Les composants sont :

- Casque de simulation virtuelle : qui permet la visualisation de la scène chirurgicale.
- Deux manettes haptiques qui contrôlent des outils chirurgicaux virtuels (des ciseaux et une manette) représenter dans la simulation grâce à des modèles.
- Un modèle de l'organe qui va subir la chirurgie « le foie », et le modèle du ciseau et de la pincette.
- Unity 3D, grâce à l'image rendering transforme l'organe, et la scène chirurgicale de la forme numérique a la forme visuelle. Il permet aussi l'ordonnancement des taches grâce au Task scheduler.
- Algorithme1 : qui effectue le calcul des déformations, et la position de l'outil, et la force de réaction de l'organe.
- Algorithme de réaction : qui permet de traduire la force en des valeurs utilisables par la manette.

Remarques : Dans la Figure 6 :

- Les objets représentés avec la couleur violette sont des objets physiques.
- Les objets représentés avec la couleur rose sont des objets 3D.
- Les objets représentés avec la couleur blanche sont des algorithmes.

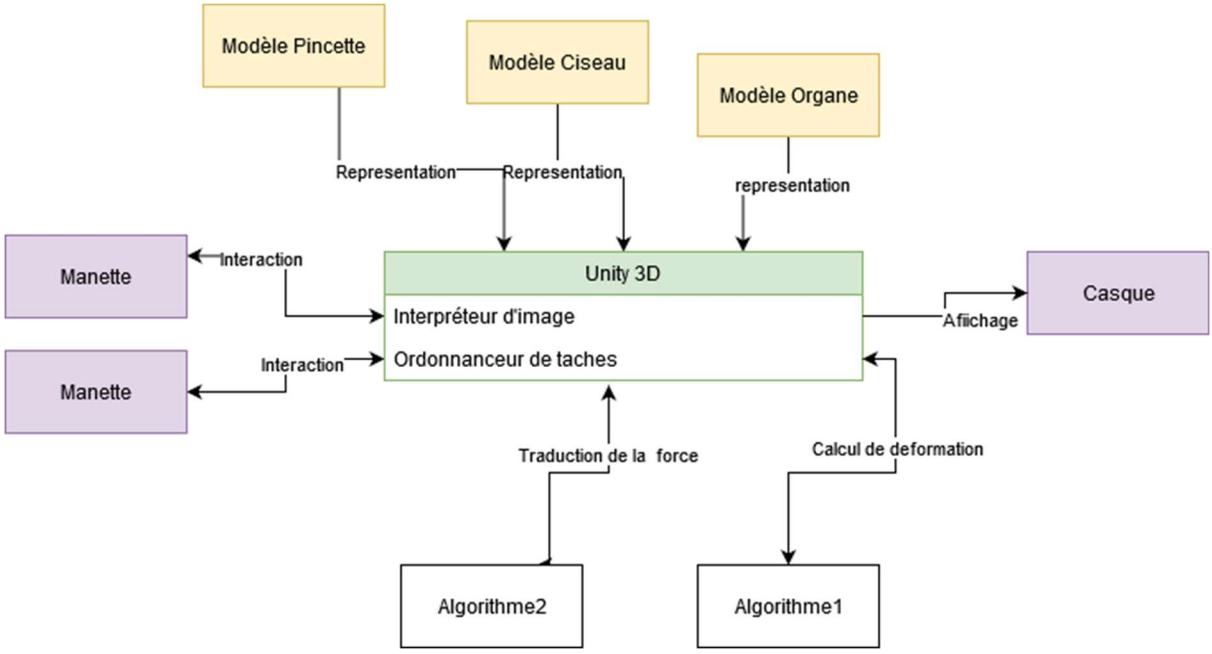


*Figure 6. Composants du système « VRAnat »*

Dans la suite, nous allons analyser les modules (objets) du système, ainsi que la définition des rôles, propriétés et les entrées/sorties de chacun.

## III.2.1 Les objets physiques

On distincts trois objets physiques dont la manette haptique, le casque VR et le moteur de jeux Unity3D qui se compose de d'autres objets physique et numérique.

- **Manette OMNI**

Le premier module interactif dans ce modèle est la manette haptique « PHANToM OMNI » montré dans la figure 9 et [27], qui permet de produire une réaction au mouvement de l'utilisateur. La manette ne peut reproduire et interpréter que des valeurs de force inférieures ou égales à 3.3 N. Elle a une résolution de 0.055 mm, ce qui implique que la manette ne détecte un mouvement avec des intervalle de 0.055 mm. La manette permet à l'utilisateur d'interagir

avec la simulation en contrôlant le mouvement des ciseaux et d'une pincette. La manette sert à interpréter la force appliquée par l'utilisateur selon sa résolution et de transmettre cette force et sa position au moteur de simulation. Ensuite, la manette permet à la simulation d'interagir avec l'utilisateur. En produisant une force de réaction afin d'affecter la perception haptique de l'utilisateur. Le Tableau 1 représente ainsi les spécifications de la manette. Utilisateur -> Simulation, lorsque l'utilisateur effectue un mouvement avec la manette, et Simulation -> Utilisateur, représente lorsque la simulation produit une force de réaction.

*Figure 7. Manette PHANToM OMNI*

*Tableau 1. Spécifications de la manette*

| Interaction | Utilisateur -> Simulation | Simulation -> Utilisateur |
|---|---|---|
| Entrées | 0<Force Utilisateur (N)<3.3 | 0.0< Force de réaction de l'organe <3.3 |
| Sorties | Position de la manette, 0.0<Force <3.3 | 0<Force de résistance (N)<3.3 |
| Résolution de la manette | 0.055 cm | |

- **Le casque VR**

Le casque VR est un objet (appareil) à poser sur la tête de l'utilisateur. Sert à visionner la scène opératoire. On ne va pas entrer dans ses spécifications, car son rôle consiste juste à retransmettre l'affichage de la simulation.[18]

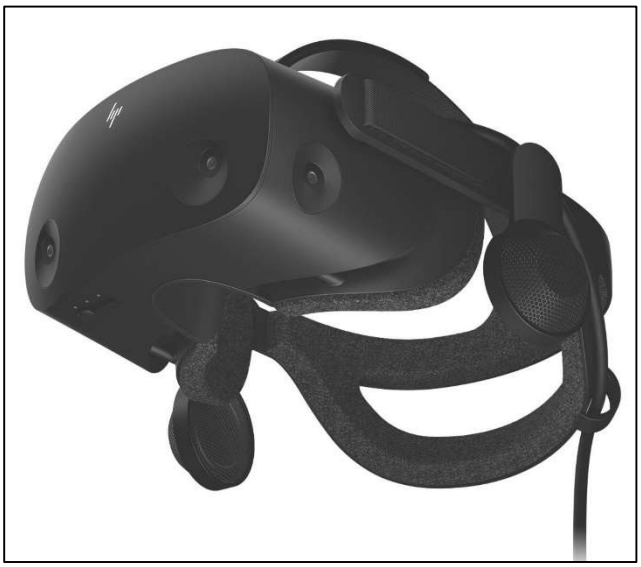

*Figure 8. Casque VR*

- **Unity3D**

L'outil "Unity 3D" est un moteur de jeux à 3 dimensions, son rôle consiste à faire le lien entre les parties matérielles et les parties logiciels du système. Il est responsable de l'affichage à travers le casque de la scène opératoire montré sur la Figure 9 ainsi que les outils chirurgicaux, et l'organe, grâce à Image Rendering depuis leurs modèles contenus dans des fichiers "XML". Qui contiennent leur structure, leur texture, et même la résistance pour l'organe. Ce module assure la communication entre les autre composant Lorsque ce module reçoit une force et une position depuis la manette, ce dernier va traduire la position de la manette en position du model virtuelle (position du model 3d du ciseaux et pincette) et les transmettre à l'Algorithme1 qui calcule la déformation à appliquer à l'organe. Ensuite, il envoi la force de réaction qui sera produite par la manette.

*Figure 9. Scène opératoire implémenté avec Unity 3D*

## III.2.2 Les objets 3D

Les modèles ou les objets 3D, sont des Plugins ou des modèles implémentés par les développeurs de ce système permettent d'obtenir la structure sous forme de fichier xml des différents objets qui interagissent durant la simulation, ci-dessous on va voir les différents modèles 3D impliqué dans ce système.

- **Organe**

Comme montré dans la Figure 10 l'organe qu'on a utilisé lors de notre étude, est une simulation du foie, décrit par sa structure une forme géométrique qui consiste en un ensemble de maille ou de point interconnecter ( sous la forme de tableaux qui contiennent les liaisons entre les points qui composent le model) , une résistance qui varie selon différentes zones dans l'organe représenté avec des couleurs et une texture, dans notre étude on a besoin juste de sa résistance et sa structure à interagir avec l'Algorithme1 qu'on va le voir par la suite.

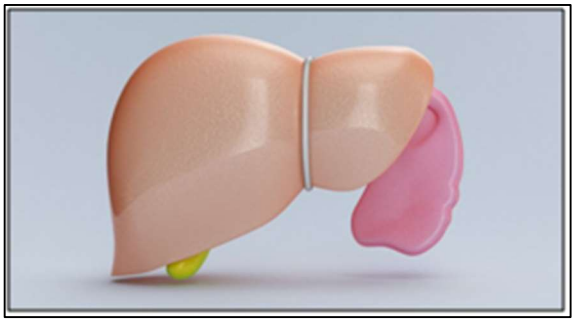

*Figure 10. Modèle 3D représentant un foie*

- **Pincettes**

La pincette est un outil qu'utilise l'utilisateur pour étirer un organe et avoir des déformations spécifiques selon le besoin du chirurgien, son modèle 3D représenté dans la Figure 11

- **Ciseaux**

Les ciseaux sont aussi un outil qu'utilise l'utilisateur pour couper un organe et avoir des déformations spécifiques selon le besoin du chirurgien, son modèle 3D représenté dans la Figure 12.

*Figure 11.. Modèle 3D représentant une pincette*

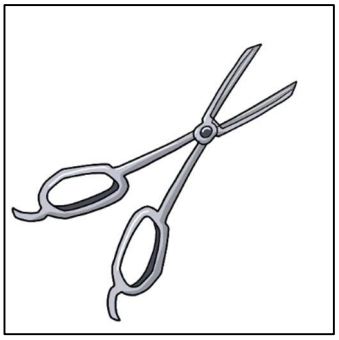

*Figure 12. Modèle 3D représentant des ciseaux*

## III.2.3 Les algorithmes

Les algorithmes sont le cœur de ce système, on distingue 2 algorithmes, dont le premier est le responsable des déformations, le deuxième sert à contrôler les réactions, ci-dessous une description détaillée sur ces deux algorithmes.

- **Algorithme1**

L'Algorithme1, est l'unité responsable des calculs des déformations de l'organe, il prend en entré la résistance de l'organe, la structure de l'organe, la position de l'outil, la force appliquée par l'utilisateur et le type d'outil.

Lorsque l'utilisateur interagie avec la simulation en utilisant la manette. Si la position de l'outil est différente de celle de l'organe alors l'Algorithme1 n'est pas exécuté et l'organe ne subit aucune déformation, Mais si l'outil est en contact avec l'organe alors algorithme 1 est exécuté.

- L'algorithme permet aussi de calculer la force de réaction de l'organe selon sa résistance et la force appliquée par l'utilisateur, qui sera traduite par Algorithme de réaction en force interprétable par l'organe.

La déformation varie selon la résistance de l'organe, sa structure, le point de l'organe qui est atteint par l'outil, la force appliquée sur l'organe et la position du point en caract. avec l'outil. Chaque sommet de l'organe est caractérisé par des intervalles de force (Fd, Fdi, Fa, Fe) qui prennent en paramètre la résistance de l'organe, la structure de l'organe, et la position du sommet.
Ainsi nous pouvons déduire les spécifications suivantes selon l'outil utilisé.

- **La pincette**

Lorsque l'utilisateur interagie avec la simulation avec la manette qui contrôle le mouvement du model de la pincette. L'organe vas subir des déformations selon les caractéristiques du point en contacte.

  - Si la force appliquée par la manette est inferieure a la force de déformation, alors ce point de l'organe ne vas subir aucune déformation.
  - Si la force appliquée est supérieur ou égale à la force de déformation mais inferieur a la force nécessaire pour agripper le point, alors le point sera déformé.
  - Si la force appliquée est supérieure ou égale à la force d'agrippement, mais inferieure a la force requise pour produire un endommagement, alors le point sera agrippé.
  - Si la force appliquée est supérieure ou égale à la force requise pour endommager le point, alors il sera endommagé.

Ce qui nous permet de déduire le Tableau 2, qui représente les spécifications de l'algorithme1 lors de l'utilisation d'une pincette sur un point de l'organe.

*Tableau 1. Résultat de Algorithme1selon les intervalles de force appliqué par une pincette*

| Force | Résultat |
|---|---|
| 0<=F<Fd | Aucun |
| Fd<=F<Fa | Déformation |
| Fa<=F<Fe | Agrippement |
| Fe<= F | Endommagement |

- **Les ciseaux :**

Lorsque l'interaction avec l'organe se fait par les ciseaux, les déformations que vas subir l'organe vont varier selon :

- Si la force appliquée est inferieure a la force de déformation alors aucune déformation ne sera effectué.
- Si la force appliquée est supérieure ou égale à la force de déformation mais inferieure a la force de dissection alors le point vas subir une déformation.
- Si la force appliquée est supérieure ou égale à la force dissection mais inferieur a la force qui provoque un endommagement du point alors il va subir une dissection.
- Si la force appliquée est supérieur ou égale à la force qui produit un endommagement alors le point vas subir un endommagement.

Ce qui nous permet de déduire le Tableau 3, qui représente les spécifications de l'alogithme1 lors de l'utilisation d'une pincette sur un point de l'organe.

*Tableau 2. Résultat de Algorithme1 selon les intervalles de force appliqué par un ciseaux*

| Force | Résultat |
|---|---|
| 0<=F<Fd | Aucun |
| Fd<=F<=Fdi | Déformation |
| Fdi<=F<Fe | Dissection |
| Fe<=F | Endommagement |

- **Algorithme de réaction**

L'Algorithme de réaction ou l'algorithme de réaction, consiste en la partie responsable de traduire la force de réaction calculée par l'Algorithme1, en force utilisable par la manette, afin de produire une réaction. Et calcule la position de l'outil dans la simulation virtuelle selon les contraintes de déplacement de la manette.

# III.3 Vérification du système

Cette section est consacrée à la conception architecturale du système, cette section est indispensable avant d'entamer la phase de la vérification ; pour cela, nous allons utiliser le diagramme de conception d'UML le diagramme de séquence, afin de modéliser et bien comprendre le comportement du système et les différentes interactions.

Le diagramme de séquence aide énormément dans la phase de la conception du système, il permet de décrire d'une manière simple et claire les interactions entre les modules du système afin de faciliter la tâche de la vérification. La Figure 13 montre de diagramme de séquence du système "VRnat". Comme décrite en dessous du diagramme.

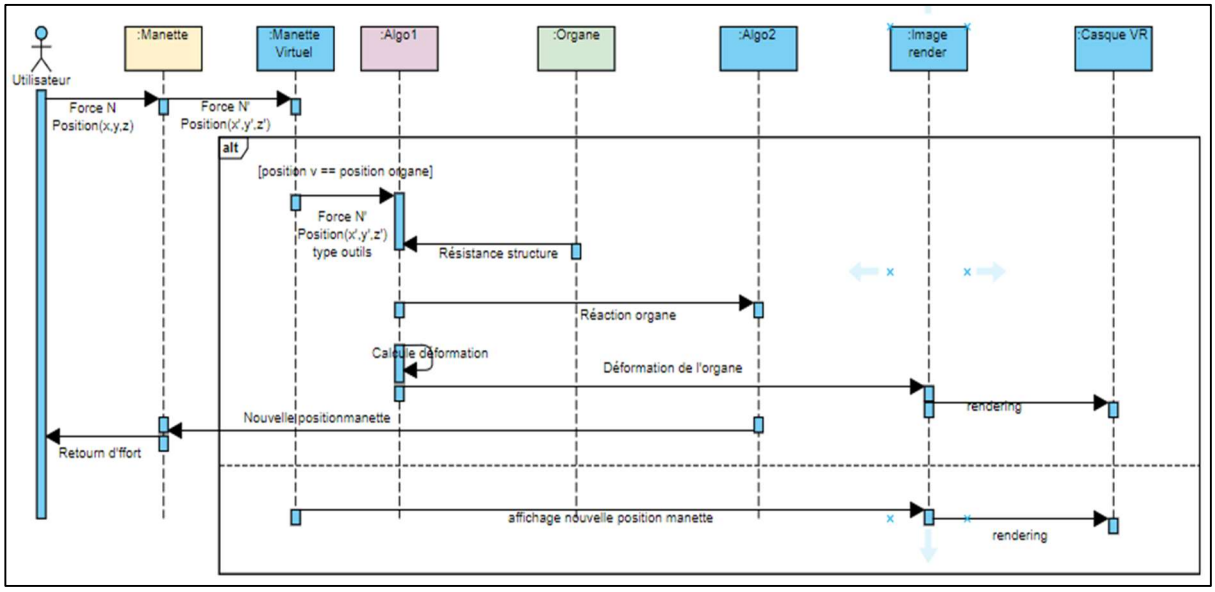


*Figure 13. Diagramme de séquence du système « VRnat »*

Dans un scénario nominal, comme montré sur la Figure 13; L'utilisateur est le point de démarrage des séquences, lorsqu'il change la position de la manette dans un espace vectoriel (x,y,z) avec une force N, la manette traduit entrées en force virtuel N' et une position virtuelle (x',y',z') dans l'environnement de simulation, si la position de la manette est égale à la position de l'organe, un appel de l'Algorithme1 se produit, l'Algorithme1 utilise la résistance de l'organe afin de calculer la déformation et la réaction.

La déformation sera envoyée à l'Image Renderer qui par conséquent affiche cette déformation sur le casque VR, la force de réaction calculer par l'Algorithme1 est alors envoyé à l'algorithme de contrôle de réaction, afin que ce dernier calcule la nouvelle position de la manette et le retour de force qui résulte de ce fait.

Sinon, si la position de l'organe n'est égale à la position de la manette, un appel de l'Image Renderer a lieu afin d'afficher le déplacement de la manette sur le casque.

## III.3.1 La mise en place de l'automate de vérification

Afin d'établir l'automate de vérification, on doit définir de façon formelle les automates du modèle checking en utilisant la structure de Kripke pour les modules du système qui sont impliqués dans la phase de la vérification ; on doit définir un ensemble d'état pour chaque automate, le passage d'un état à l'autre est une transition de paramètres partagés entre états telle que la position, la force et le type de la manette.

Les figures 14,15,16,17 représentent l'application de la structure de Kripke en automate, les états sont décrits en cercles bleu, l'état initial représenté avec double cercles, les labels des états sont à coté de cercle, les arcs de transition sont représentés avec des flèches et une description textuelle, lorsqu'une transition attend une valeur en paramètre, la description textuelle de la transition est décrite avec un "?", en cas d'envois de paramètre la description est décrite par "!" , le symbole "C" dans l'état signifie que la transition précédente doit passer par cet état.

- **La manette**

Lorsque l'utilisateur effectue un mouvement avec la manette, elle interprète la force appliquée par l'utilisateur et la position de la manette selon la position de la résolution du mouvement qui consiste en 0.055 mm (5 dans la simulation).

Ensuite la manette envoie les valeurs de position de la manette et de force vers Unity 3d. puis, elle effectue une réaction selon la force, et la position calculée par Alogithme2. Comme démontré dans la Figure 14.

La structure de Kripke est définie pour la manette sous la forme ci-dessous :

Manette = (S, s1, R, L) avec :

S est l'ensemble des états

S= {S1, S2, S3, S4, S5, S6}

S1 état initial

R sont les arcs qui relient les états

R = {(s1,s2), (s2,s3), (s2,s3), (s2,s3), (s3,s4) ,(s4,s5), (s4,s6)}

L es le label dans lequel on met une appellation pour l'état

L(S1) = {attente},

L(S2) = {mouvement_utilisateur}

L(S3) = {traduction_effort},

L(S4) = {effort_manette_traduit}

L(S5) = {attente_reaction},

L(S6) = {produire_reaction}

L(S7) = {deplacer}

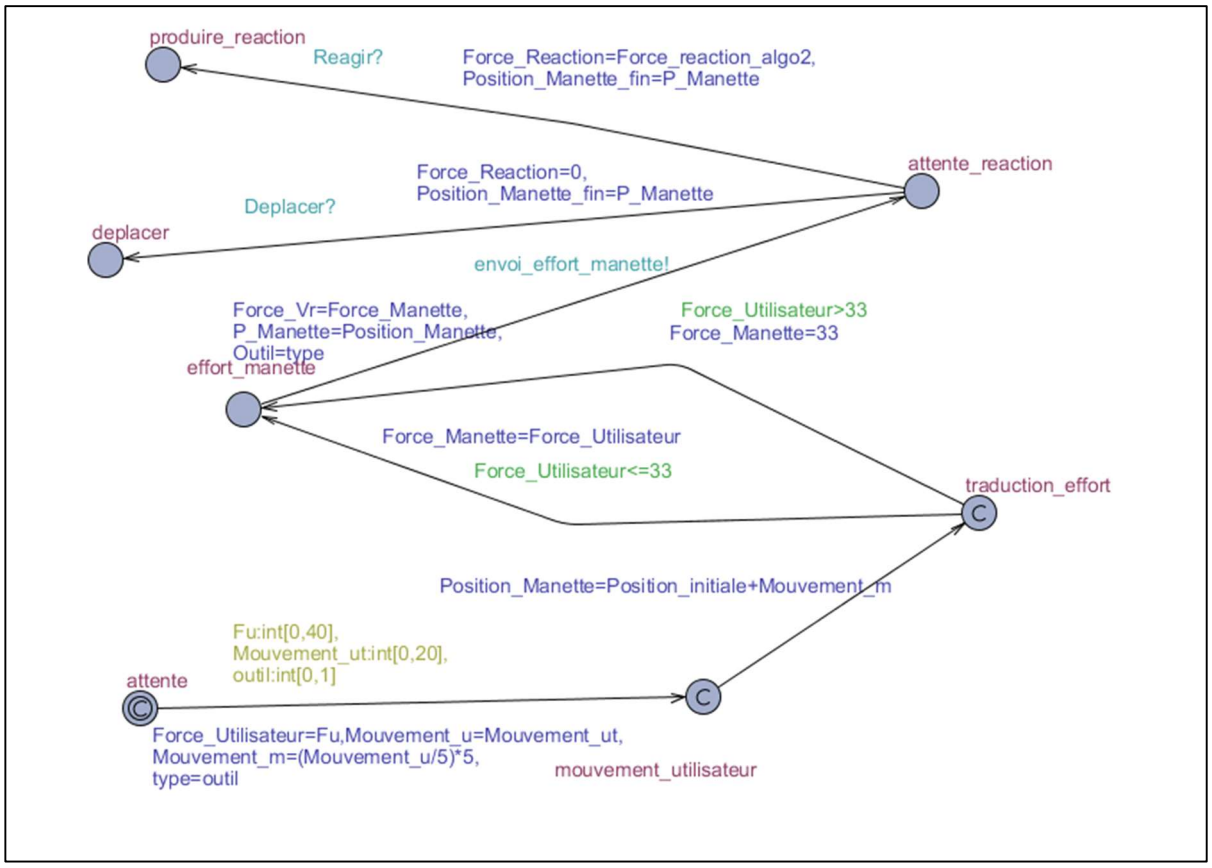


*Figure 14. Automate représentant le comportement du module « Manette »*

- **Unity 3D**

Unity 3D reçoi la position de la manette et la force appliqué, et le type d'outil. Puis, elle transforme la position de la manette en position virtuelle de l'outil chirurgical.

La manette vérifie si la position de l'outil chirurgical est en contact avec des sommets de l'organe, elle exécute alorithme1 puis elle effectue la déformation comme calculé par Algorithme1, ensuite effectue un déplacement de l'outil chirurgical selon la position calculée par algorithme de réaction, et traduit cette position de l'outil en position de la manette. Enfin

elle envoie la position finale de la manette et force de réaction à produire calculé par algorithme de réaction.

Si la position virtuelle de l'outil n'est pas en contact avec des sommets de l'organe, alors elle effectue un simple déplacement de là l'outil dans la simulation et de la manette.

La structure de Kripke est définie pour Unity3D sous la forme ci-dessous :

Unity3D= (S, s1, R, L) avec:

S est l'ensemble des états

S= {S1, S2, S3, S4, S5, S6, S7, S8, S9}

S1 état initial

R sont les arcs qui relient les états

R = {(s1,s2), (s2,s3), (s3,s4) ,(s4,s5), (s4,s6), (s4,s7), (s7,s8), (s8,s1), (s5,s9), (s6,s9)}

L es le label dans lequel on met une appellation pour l'état

L(S1) = {attente},

L(S2) = {calcule_position_outil},

L(S3) = {choisir_type_mouvement},

L(S4) = {execusion_deformation}

L(S5) = {attente_deformation}

L(S6) = {attente_reaction}

L(S7) = {voir_si_aucune_deformation}

L(S8) ={aucune_deformation}

L(S9) = {fin}

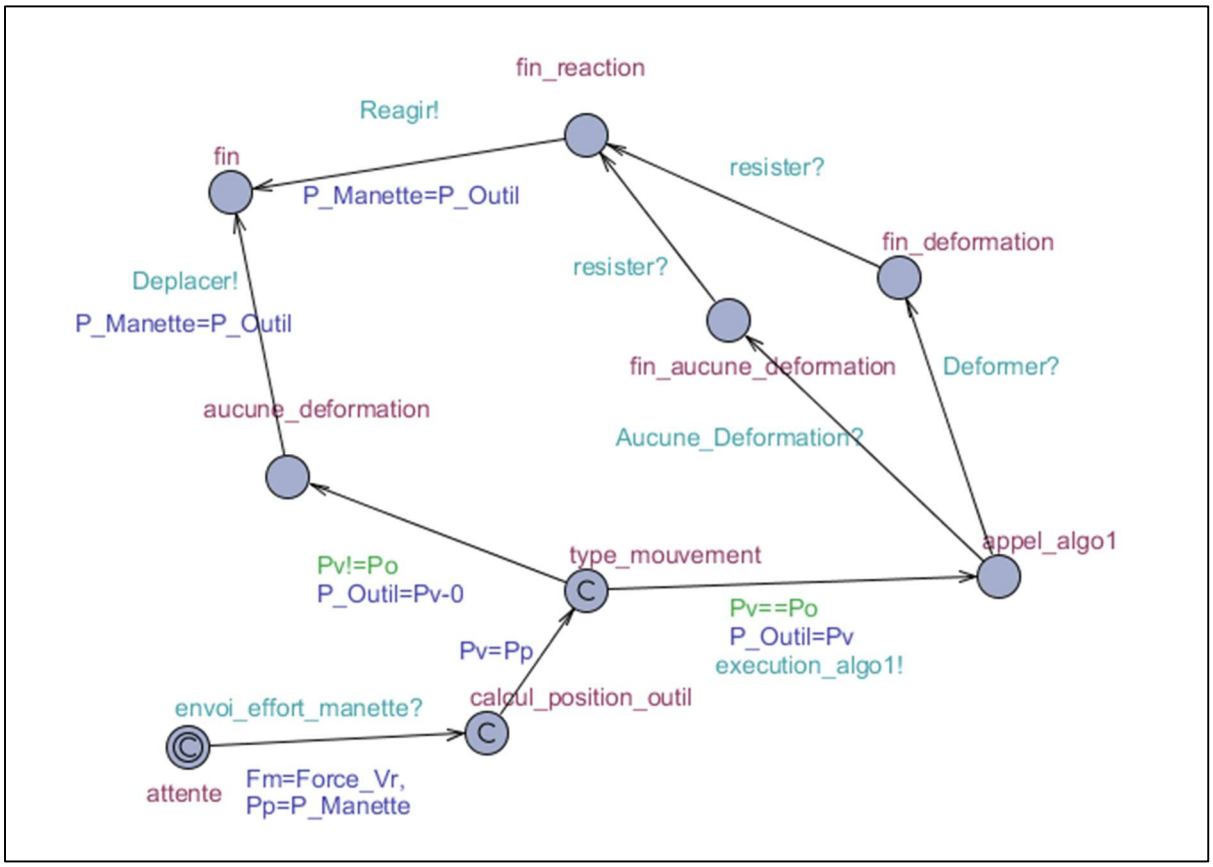


*Figure 15.Automate représentant le comportement du module « Unity 3D »*

- **Algorithme1**

Lorsque Unity 3D exécute Algorithme1, il prend en paramètre la force appliquée par la manette, la position de l'outil, le type d'outil, et la résistance de l'organe. Ensuite, il calcule la force de réaction de l'organe selon la force appliquée par la manette, et la résistance de l'organe.

Puis, l'Algorithme de réaction calcule la déformation de l'organe qui sera :

- Aucune déformation : si la force appliquée varie entre 0 et Fd qui est à valeur de force minimale requise pour effectuer une déformation.
- Une Déformation : si le type de l'outil est une pincette la force appliquée varie entre Fd et Fa Fe qui représente la force minimale pour effectuer un agrippement des sommets avec l'outil., ou si le type de l'outil est des ciseaux et la force appliquée est entre Fd et Fdi. Où Fdi est considérée la force minimale pour produire une dissection.
- Un agrippement : si l'outil est de type pincette la force appliquée varie entre Fa et Fe qui est la force minimale pour produire un endommagement.

- Une dissection : si le type de l'outil sont des ciseaux et que la force utilisée varie entre Fdi, Fe.
- Un endommagement : si la force appliquée est supérieure ou égale a Fe.

Comme représenté dans la Figure 16.

La structure de Kripke est définie pour l'Algorithme1 sous la forme ci-dessous :

Algo_déformation= (S, s1, R, L) avec :

S est l'ensemble des états

S= {S1, S2, S3, S4, S5, S6, S7, S8, S9}

S1 état initial

R sont les arcs qui relient les états

R = {(s1,s2), (s2,s3), (s3,s4) ,(s4,s5), (s4,s6), (s4,s7), (s4,s8), (s4,s9), (s5,s9), (s6,s9),(s7,s9), (s8,s9)}

L es le label dans lequel on met une appellation pour l'état

L(S1) = {attente},

L(S2) = {appelle_algo2},

L(S3) = {calcul}

L(S4) = {aucune_deformation}

L(S5) = {organe_aggripé}

L(S6) = {organe_dissectionné}

L(S7) ={organe_endommagé}

L(S8) = {organe_déformé}

L(S9) = {fin}

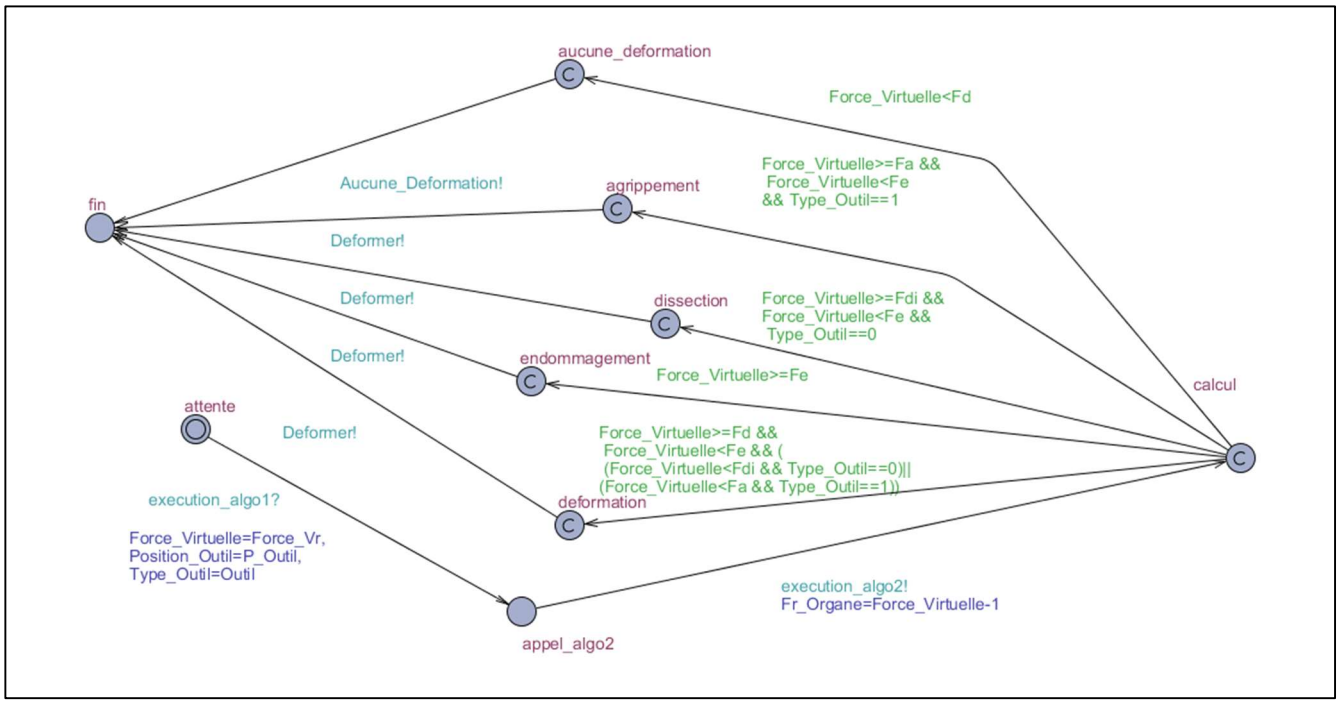


*Figure 16. Automate représentant le comportement de « Algorithme1 »*

- **Algorithme de réaction**

L'Algorithme de réaction adapte la force de réaction envoyée par l'algorithme de réaction pour qu'elle soit applicable par la manette, et calcule la position de l'outil. La Figure 17. Algorithme représentant le module « Algorithme de réaction» représente l'automate qui simule le comportement de l'Algorithme de réaction.

La structure de Kripke est définie pour la manette sous la forme ci-dessous :

Algo_reaction= (S, s1, R, L) avec:

S est l'ensemble des états

S= {S1, S2, S3, S4}

S1 état initial

R sont les arcs qui relient les états

R = {(s1,s2), (s2,s3), (s3,s4) ,(s4,s1)}

L es le label dans lequel on met une appellation pour l'état

L(S1) = {attente},

L(S2) = {calcule_force},

L(S3) = {execution_reaction},

L(S4) = {fin}

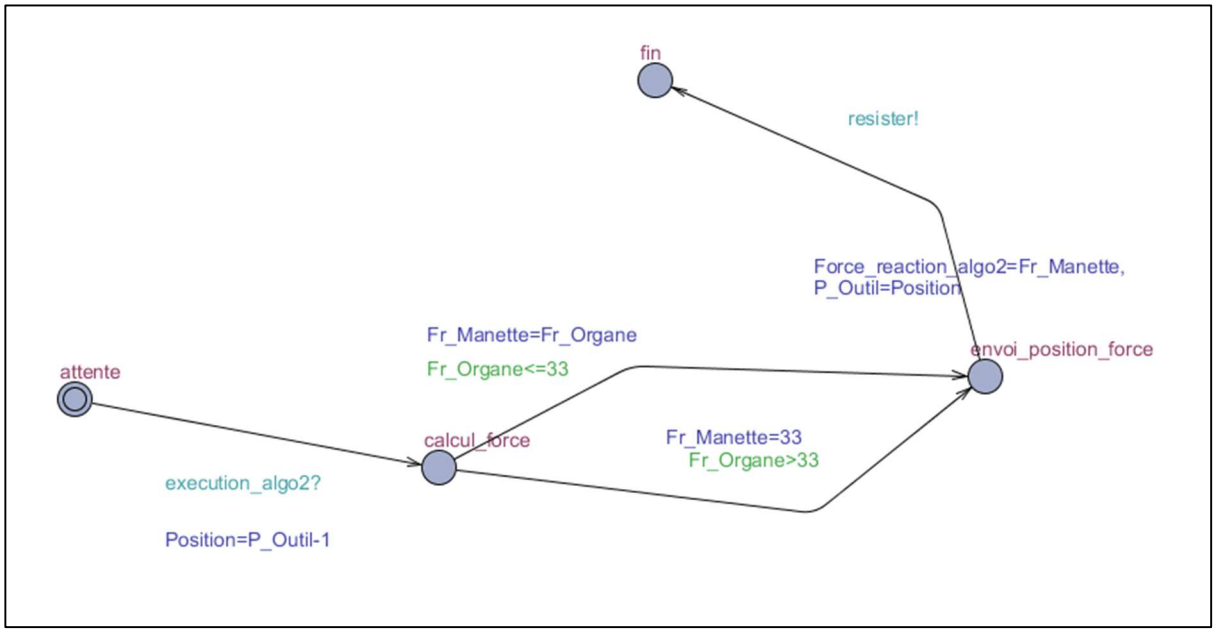


*Figure 17. Algorithme représentant le module « Algorithme de réaction»*

## III.3.2 La définition des propriétés

Afin de vérifier le système, Il faut en premier lieu extraire les propriétés du système en se basant sur les spécifications citées dans la section III.2. Ce qui nous permet de les classifier selon le module concerné.

L'étape de la vérification et la validation repose sur la vérification des automates, l'outils UPPAAL nous permet de vérifier les automates d'une manière automatique il suffit de décrire les propriétés à vérifier avec le langage de la logique temporelle, le compilateur de l'outils va vérifier si une propriété est satisfaite ou pas.

- **La manette**

Lorsque l'utilisateur effectue un mouvement, avec la manette. Elle calcule la force exercée par l'utilisateur, et la position de la manette selon sa résolution (0.055).

Ce qui implique que le degré de précision de la manette est de 0.055, ainsi la manette n'interprète que les mouvements en valeurs multiples de 0.055. Ainsi, le mouvement de la manette est le multiple de 0.055 le plus proche du mouvement de l'utilisateur.

La valeur maximale interprétable par la manette est de 3.3. Ainsi si la force appliquée par l'utilisateur ou si la force appliquée par la manette en cas de réaction est supérieure à 3.3, la manette enverrait une force égale à 3.3 grâce à l'Algorithme de réaction,

*Tableau 3. Les propriétés du module "manette"*

| Propriétés | |
|---|---|
| 1 | Quel que soit la force appliquer par l'utilisateur, la force de la manette est inférieur ou égale à 3.3N. |
| 2 | La manette interprète la force de l'utilisateur en force manette. |
| 3 | La position de la manette = La position actuelle + mouvement de la manette. |
| 4 | Quel que soit x, Le mouvement de la manette = 0.055*x <= (mouvement de l'utilisateur) <0.055*(x+1) |
| 5 | La manette transmet la position et la force vers Unity 3D. |

- **Unity 3D**

Lorsque la manette effectue un mouvement, le moteur de simulation Unity 3D traduit la position de la manette en position de l'outil virtuelle (ciseaux ou pincette). Puis, si la position virtuelle n'est pas en contact avec l'organe, alors aucune déformation n'est effectuée sur l'organe. Mais si l'outil est en contact avec l'organe, alors l'Algorithme1est exécuté.

Lorsque Algorithme1 est exécuté, grâce à l'image renderer de Unity 3D déforme l'organe, et déplace l'outil. Puis, le moteur, transmet la force de réaction traduite par Algorithme de réactionà la manette afin de produire une force de réaction.

*Tableau 4. Les propriétés du module "Unity 3D"*

| Propriété | |
|---|---|
| 6 | La position de la manette est traduite en position virtuelle. |
| 7 | Si la position de l'outil est égale à la position de l'organe, l'Algorithme1est exécuté. |
| 8 | Si la position de l'outil est égale à la position de l'organe, Unity 3D transmet la force de réaction à la manette. |
| 9 | Si la position de l'outil est égale à la position de l'organe, Unity 3D effectue des déformations |
| 10 | Si la position de l'outil est différente de la position de l'organe, Aucune déformation n'est effectuée. |

- **Algorithme1**

Lorsque l'Algorithme1 est exécuté il prend en paramètre la force de la manette et la résistance de l'organe, la structure de l'organe, et la position du point de l'organe qui est en contact avec l'outil, et le type de l'outil, comme expliqué dans III.4.4, puis les transmet à Unity 3D. De ce fait, les forces de Dissection, Déformation, Endommagement, Agrippement varient selon la résistance de l'organe, La structure de l'organe, et la position du point de l'organe. Ce qui implique :

L'Algorithme de réaction calcule la force de réaction de l'organe et l'envoi a l'Algorithme de réaction qui la traduit en force utilisable par la manette.

Le Tableau 5 permet de regrouper les propriétés de l'Algorithm1

*Tableau 5.. La représentation des propriétés du module "Algorithme1"*

| Propriétés | |
|---|---|
| 11 | Si la force appliquée est inferieure a Fd, et quel que soit l'outil utilisé, aucune déformation ne sera appliquée. |
| 12 | Si la force appliquée varie entre [Fd,Fdi] et le type d'outil est un ciseaux, alors l'Algorithme1 calcule une déformation |
| 13 | Si la force appliquée varie entre [Fd,Fa] et le type d'outil est une pincette, alors l'algorithme calcule une déformation. |
| 14 | Si la force appliquée varie entre [Fdi,FE] et le type d'outil est un ciseaux, alors l'algorithme calcule une dissection. |
| 15 | Si la force appliquée varie entre [Fa,Fe] et le type d'outil est une pincette, alors l'algorithme calcule un agrippement. |
| 16 | Si la force appliquée est supérieure à Fe, et quel que soit le type d'outil, alors l'organe subit un endommagement. |
| 17 | La réaction de l'organe varie selon la force appliquée et la résistance de 'organe |

- **Algorithme de réaction**

L'algorithme de réaction assure sert à adapter la force de réaction calculée par l'Algorithme de réaction en force reproduisible par la manette [0,3.3].

*Tableau 6.. La représentation des propriétés du module « Algorithme de réaction »*

| Propriétés | |
|---|---|
| 19 | Si la force appliquée est supérieure à 3.3N, La force est traduite a 3.3N |

### III.3.3 Le rapport de tests

Pout la vérification de l'automate, nous avons utilisé Uppaal qui permet de vérifier les propriétés du système selon les automates générés comme démontré dans les figures 14, 15, 16 et 17. Mais afin de parvenir à utiliser l'outil qui ne prend pas les valeurs doubles ou floats, nous avons modifier les intervalles de force de la manette [0,3.3] a [0,33]. En ce qui concerne les forces applicables par l'utilisateur nous les avons limitées à 40. Pour la résolution du mouvement de la manette nous avons utilisé une résolution égale à 5 au lieu de 0.055, durant la vérification, nous effectuons les tests de spécifications en simulant la position dans une seule dimension : la position de l'organe est 0 et la position d'un seul sommet de l'organe qui est 5, et les valeurs de forces : Fd=10, Fdi=19, Fa=21, Fe=29.

Puis nous avons utiliser l'interface de vérification de Uppaal qui permet de vérifier si des propriétés de vivacité et de suretés sont satisfaites dans le système et ce a travers des requêtes basées sur la logique temporelle et les prédicats comme démontré dans la figure 18 qui représente les deux formules de vérification pour les propriétés de suretés, et la figure 19 qui présente les méthodes de vérifications des propriétés de vivacité.

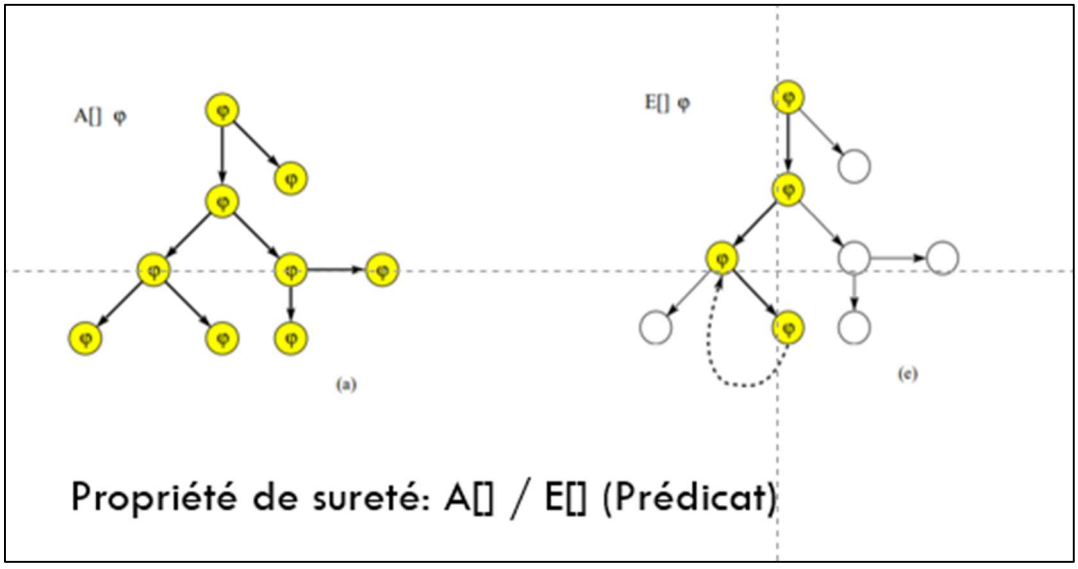


*Figure 18. Formules de vérification des propriétés de sureté dans Uppaal*

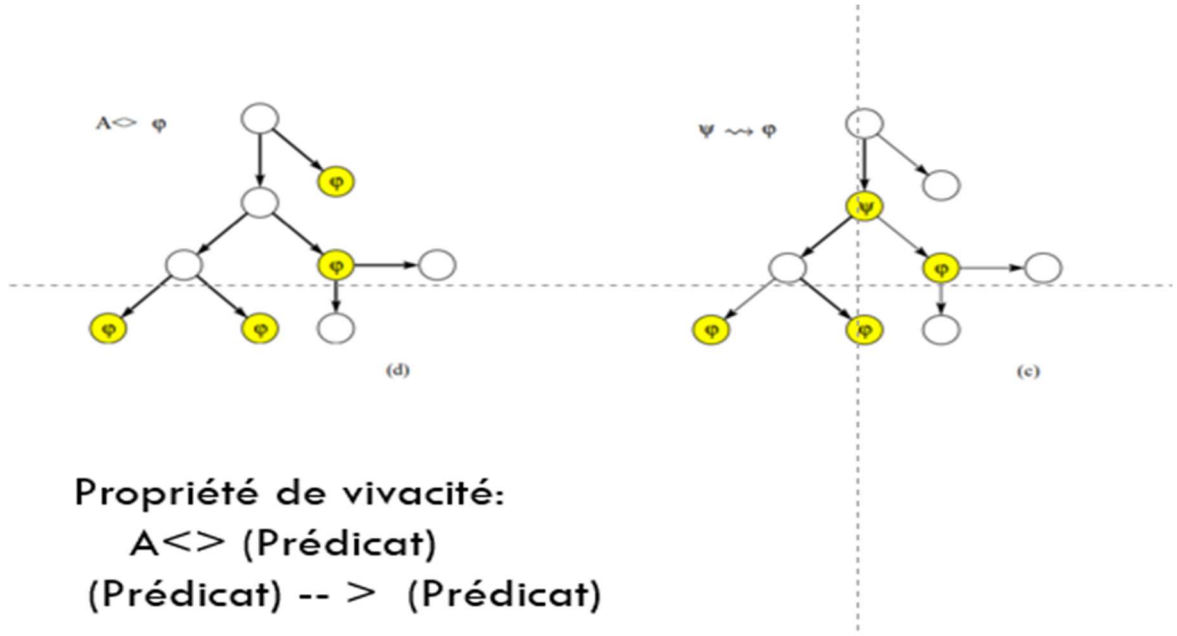


*Figure 19. Formules de vérification des propriétés de vivacité dans Uppaal*

### III.3.3.1 La manette

La mise en place des vérifications des propriétés déduites dans la section sur l'automate qui représente le module Manette.

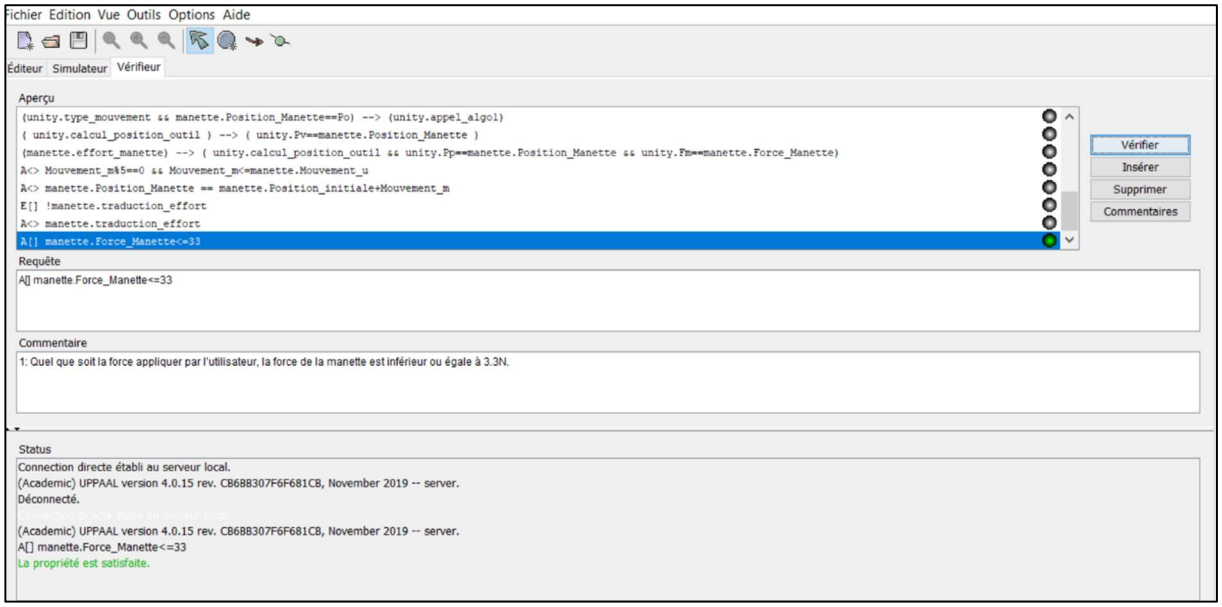


*Figure 20. Vérification de la propriété 1*

*Tableau 7. Vérification de la propriété 1*

| N | 1 |
|---|---|
| Propriétés | Quel que soit la force appliquée par l'utilisateur, la force de la manette est inférieure ou égale à 33. |
| Logique temporelle | A[] manette.Force_Manette<=33 |
| Résultat | Valide |

Le Tableau 7. Vérification de la propriétés 1 implique que peu importe la force utilisateur appliquée, la force utiliser par la manette sera toujours inferieur ou égale 33 (3.3N) comme vérifier à travers Uppaal comme démontré dans la figure 20.

*Tableau 8. Vérification de la propriété 2*

| N | 2 |
|---|---|
| Propriétés | La manette interprète la force de l'utilisateur en force manette. |
| Logique temporelle | A<> manette.traduction_effort |
| Résultat | Valide |

La propriété vérifiée dans le Tableau 8, signifie que pour tout l'état de l'automate la manette traduit la force de l'utilisateur.

*Tableau 9. Vérification de la propriété 3*

| N | 3 |
|---|---|
| Propriétés | La position de la manette = La position actuelle + mouvement de la manette. |
| Logique temporelle | A<> manette.Position_Manette == manette.Position_initiale+manette.Mouvement_m |
| Résultat | Valide |

Comme représenté dans le Tableau 9, pour tous cas la position de la manette prend en paramètre la position initiale de la manette et le mouvement de la manette.

*Tableau 10. Vérification de la propriété 4*

| N | 4 |
|---|---|
| Propriétés | Quel que soit x, Le mouvement de la manette = 5*x <= (mouvement de l'utilisateur |
| Logique temporelle | A<> manette. Mouvement_m%5==0 && manette.Mouvement_m<=manette.Mouvement_u |
| Résultat | Valide |

Le Tableau 10 représente la propriété qui consiste en : la valeur du mouvement de la manette doit être un multiple de 5 et qu'il doit être inferieur ou égale au mouvement de l'utilisateur.

*Tableau 11. Vérification de la propriété 5*

| N | 5 |
|---|---|
| Propriétés | La manette transmet la position et la force vers Unity 3D. |
| Logique temporelle | (manette.effort_manette) --> ( unity.calcul_position_outil && unity.Pp==manette.Position_Manette && unity.Fm==manette.Force_Manette) |
| Résultat | Valide |

La propriété représentée dans le Tableau 11 consiste en lorsque la manette produit un mouvement, elle envoi a Unity 3D la position de la manette et la force de la manette.

Les tableaux suivants représentent les vérifications des propriétés qui caractérisent le module de Unity 3D.

*Tableau 12. Vérification de la propriété 6*

| N | 6 |
|---|---|
| Propriétés | La position de la manette est traduite en position virtuelle. |
| Logique temporelle | ( unity.calcul_position_outil ) --> ( unity.Pv==manette.Position_Manette ) |
| Résultat | Valide |

Le Tableau 12 signifie que si le module est dans l'état calcul position outil, la position de l'outil est égale à la position de la manette.

*Tableau 13. Vérification de la propriété 7*

| N | 7 |
|---|---|
| Propriétés | Si la position de l'outil est égale à la position de l'organe, Algorithme1 est exécuter. |
| Logique temporelle | (unity.type_mouvement && manette.Position_Manette==Po) --> (unity.appel_algo1) |
| Résultat | Valide |

Dans le Tableau 13, si l'automate de Unity 3D est dans l'état type_mouvement et la position de la manette est en contact avec la position de l'organe, alors Unity 3D exécute Algorithme1.

*Tableau 14. Vérification de la propriété 8*

| N | 8 |
|---|---|
| Propriétés | Si la position de l'outil est égale à la position de l'organe, Unity 3D transmet la force de réaction à la manette. |
| Logique temporelle | A[] (unity.fin_reaction ) imply ( unity.Pv==Po ) |
| Résultat | Valide |

Lorsque Unity 3D est dans l'état fin_reaction alors la position de l'outil doit être égale a la position de l'outil (Pv) comme vérifier dans Tableau 14.

*Tableau 15. Vérification de la propriété 9*

| N | 9 |
|---|---|
| Propriétés | Si la position de l'outil est égale à la position de l'organe, Unity 3D effectue des déformations |
| Logique temporelle | A[] ( unity.fin_reaction) imply ( manette.Pv==Po ) |
| Résultat | Valide |

Dans le Tableau 15, Si l'automate est dans l'état fin_reaction alors la position de l'outil doit être égale à la position de l'organe.

*Tableau 16. Vérification de la propriété 10*

| N | 10 |
|---|---|
| Propriétés | Si la position de l'outil est différente de la position de l'organe, Aucune déformation n'est effectuée. |
| Logique temporelle | A[] ( unity.aucune_deformation) imply ( manette.Pv!=Po ) |
| Résultat | Valide |

Le Tableau 16 signifie que si dans Unity l'état aucune_deformation est atteint alors la position virtuelle de l'outil est différente de la position de l'organe.

- **L'Algorithme1**

Cette partie représente les vérifications des propriétés du module Algorithme1

*Tableau 17. Vérification de la propriété 11*

| N | 11 |
|---|---|
| Propriétés | Si la force appliquée est inferieur a Fd, et quel que soit l'outil utilisé, aucune déformation ne sera appliquée. |
| Logique temporelle | ( Algorithme1.calcul && manette.Force_Manette<algorithme1.Fd ) --> ( algorithme1.aucune_deformation ) |
| Résultat | Valide |

Le Tableau 17 signifie que si l'algorithme est dans l'état calcul et que la force appliquée est inferieur a la force de déformation alors pout importe le type de la manette. L'Algorithme1 calcule aucune déformation.

*Tableau 18. Vérification de la propriété 12*

| N | 12 |
|---|---|
| Propriétés | Si la force appliquée varie entre [Fd,Fdi] et le type d'outil est des ciseaux, alors algorithme1 de déformation calcule une déformation |
| Logique temporelle | (Algorithme1.calcul && manette.Force_Manette>=algorithme1.Fd && manette.Force_Manette<algorithme1.Fe && manette.Position_Manette==Po && manette.Force_Manette<algorithme1.Fdi && manette.type==0 ) --> ( algorithme1.deformation ) |
| Résultat | Valide |

Le Tableau 18 représente que si la force appliquée varie entre Fd et Fdi et que le type de la manette est ciseaux (manette.type==0) alors la déformation calculée est une déformation.

*Tableau 19. Vérification de la propriété 13*

| N | 13 |
|---|---|
| Propriétés | Si la force appliquée varie entre [Fd,Fa] et le type d'outil est un ciseaux, alors Algorithme1 calcule une déformation |
| Logique temporelle | (Algorithme1.calcul && manette.Force_Manette>=algorithme1.Fd && manette.Position_Manette==Po && manette.Force_Manette<algorithme1.Fa && manette.type==1 ) --> ( algorithme1.deformation ) |
| Résultat | Valide |

Le Tableau 19 représente que si la force appliquée varie entre Fd et Fa et que le type de la manette est pincette (manette.type==1) et que l'automate est dans l'état calcul. Alors la déformation calculée est une déformation.

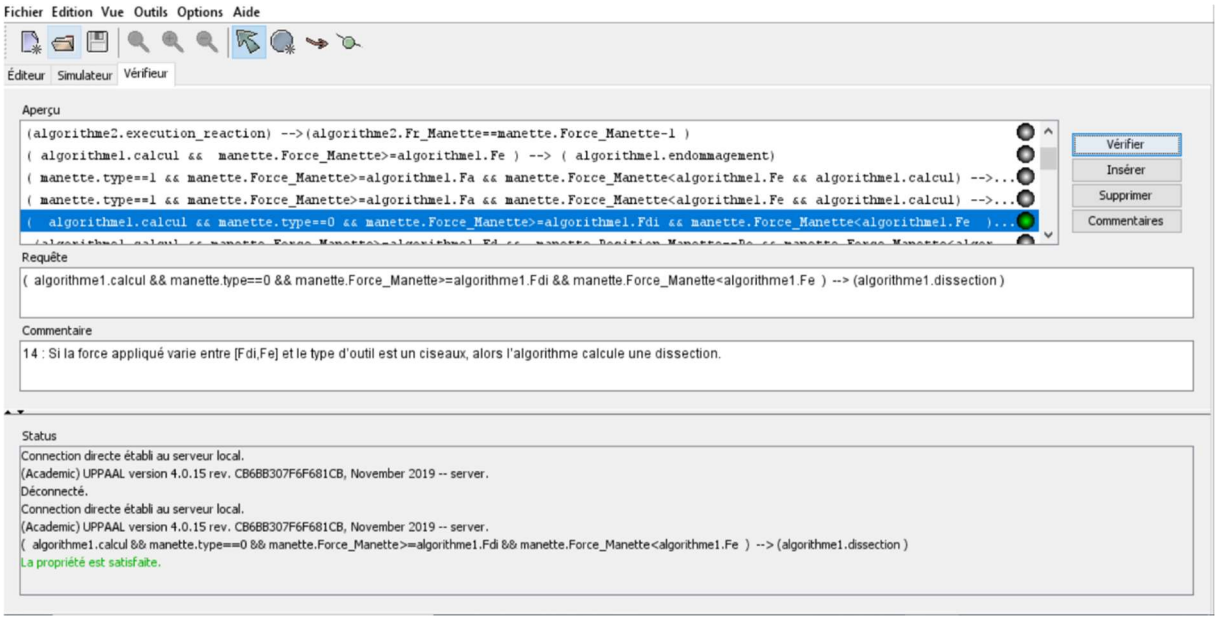


*Figure 21. Vérification de la propriété 14*

*Tableau 20. Vérification de la propriété 14*

| N | 14 |
|---|---|
| Propriétés | Si la force appliqué varie entre [Fdi,Fe] et le type d'outil est un ciseaux, alors l'algorithme calcule une dissection. |
| Logique temporelle | ( Algorithme1.calcul && manette.type==0 && manette.Force_Manette>=agorithme1.Fdi && manette.Force_Manette<algorithme1.Fe ) --> (algorithme1.dissection ) |
| Résultat | |

Le Tableau 20 et la Figure 21 représente que si la force appliquée est supérieure ou égale à Fe et peu importe le type de manette et que l'automate est dans l'état calcul. Alors la déformation calculée est une dissection.

*Tableau 21. Vérification de la propriété 15*

| N | 15 |
|---|---|
| Propriétés | Si la force appliqué varie entre [Fd,Fa] et le type d'outil est une pincette, alors l'algorithme calcule une déformation |
| Logique temporelle | ( manette.type==1 && manette.Force_Manette>=Algorithme1.Fa && manette.Force_Manette<Algorithme1.Fe && Algorithme1.calcul) --> Algorithme1.agrippement |
| Résultat | Valide |

Le Tableau 21 Implique que si la force appliquée varie entre Fd et Fa et que le type de manette0

Est pincette et que l'état dans l'automate dans Algorithme1 est calcul. Alors l'algorithme calcule un agrippement.

*Tableau 22. Vérification de la propriété 16*

| N | 16 |
|---|---|
| Propriétés | Si la force appliquée est supperieur ou egale a la force d'endommegement |
| Logique temporelle | ( manette.type==1 && manette.Force_Manette>=algorithme1.Fa && manette.Force_Manette<algorithme1.Fe && agorithme1.calcul) --> algorithme1.agrippement |
| Résultat | Valide |

Dans le Tableau 22 si la force appliquée est supérieure ou égale a Fe et quel que soit le type de manette et que l'état dans l'automate dans Algorithme1 est calcul. Alors l'algorithme calcule un endommagement.

*Tableau 23. Vérification de la propriété 17*

| N | 17 |
|---|---|
| Propriétés | La réaction de l'organe varie selon la force appliquée et la résistance de l'organe |
| Logique temporelle | (algorithme de réaction.execution_reaction) -->(algorithme de réaction.Fr_Manette==manette.Force_Manette-Resistance |
| Résultat | Valide |

Dans le Tableau 23 Si l'automate est dans l'état execution_reaction alors la force de réaction de la manette varie selon la force de la manette et la résistance.

*Tableau 24. Vérification de la propriété 18*

| N | 18 |
|---|---|
| Propriétés | Si la force applique est supérieure à 3.3N, La force est traduite a 3.3N |
| Logique temporelle | (algorithme de réaction.calcul_force) --> (algorithme de réaction.Fr_Manette<=33) |
| Résultat | Valide |

Dans le Tableau 24 Si l'automate est dans l'état calcul_force alors la valeur de la force de réaction de la manette doit être inferieur a 33.

# III.4 Discussion des résultats des tests

A travers le rapport de tests dans la section présidente, nous pouvons conclure que :

Les propretés de sureté qui consiste en 1,3,4,6,17 et 18 sont toujours respectés dans le système étudié et ce peu importe la force de l'utilisateur, le type de manette, ou la position.

Les propriétés de vivacité qui sont les propriétés 2,5,7,8,9,10,11,12,13,14,15, et 16 sont vérifiés dans le système selon les forces appliquées, la position de la manette, le type d'outil, et aussi par la résistance de l'organe, sa structure, et la position de la zone de l'organe en contact avec la manette qui affectes les valeurs de Fd, Fa, Fdi, Fe.

# Conclusion Générale

Notre étude a consisté en premier lieu à analyser des travaux de vérifications antérieurs appliqués sur des systèmes critiques surtout sur ceux basé sur la réalité virtuelle ce qui nous a mené à choisir la vérification à travers les méthodes formelles qui permettent de vérifier et trouver des failles dans les systèmes complexes et critiques comme le cas du système de notre étude.

Comme outil de vérifications nous avons choisi un outil de vérification automatique qui permet de reproduire le système sous forme d'automates basés sur le model checking puis de vérifier que les propriétés fondamentales qui caractérises le système sont satisfaites dans ces automates.

Afin d'effectuer la vérification nous avons dû extraire les spécifications fonctionnelles du système « VRAnat ». Afin de pouvoir s'en servir comme base des propriétés à vérifier dans le système. Nous avons noté quelques propriétés qui pourraient avoir un impact critique sur la sureté du système, nous citons parmi elles :

Les propriétés 11, 12, 13, 14, et 15 : qui ont un impact direct sur le calcul de la déformation par l'Algorithme1. Car, en cas de différences dans ces calculs les déformations appliquées par Unity sur l'organe ne seront plus fiables.

Les propriétés 1 et 18 : qui impliquent que toute force appliquée par l'utilisateur, ou en réaction de l'organe qui dépasse les 3.3 N sera considérée comme une force égale à 3.3N. Ceci implique les contraintes suivantes :

- La manette ne produira pas une force de réaction force supérieur à 3.3N, et ce peu importe la force appliquée par le mouvement de l'utilisateur. Une restriction dans le choix des chirurgies qui pourraient être simulait dans le système, surtout celles qui nécessitent l'exercerions de grande force comme celles sur les os.

Ainsi, la vérification des propriétés cités précédemment est essentielle pour vérifier la sureté du système pour chaque extension ajoutée. Aussi, pour intensifier la vérification du système il serait plus intéressant d'effectuer des tests de spécification plus approfondies, surtout pour les propriétés qui concernent l'Algorithme1 : 11, 12 ,13, 14, et 15. Et de prendre en considération les contraintes temporelles qui caractérisent tous les systèmes temps réel, qui peuvent être vérifiés avec les divers outils de vérifications basés sur le model-checking, parmi eux UPPAAL.

# Annexe

- **L’outil UPPAL**

UPPAAL est une boîte à outils puissante pour la vérification et la validation via la simulation graphique et la vérification automatique du modèle des systèmes en temps réel.

Il est composé de deux parties principales :

–une interface utilisateur graphique (GUI) (exécutée sur le poste de travail des utilisateurs) montré dans la Figure 22, Figure 23 et la Figure 24.

–un moteur de vérification de modèle (exécuté par défaut sur le même ordinateur que l'interface utilisateur, mais peut également s'exécuter sur un serveur plus puissant).

Il a été développé conjointement par l'Université d'Uppsala en Suède et l'Université Aalborg au Danemark, la dernière version est 4.0.15 (novembre 18, 2019)

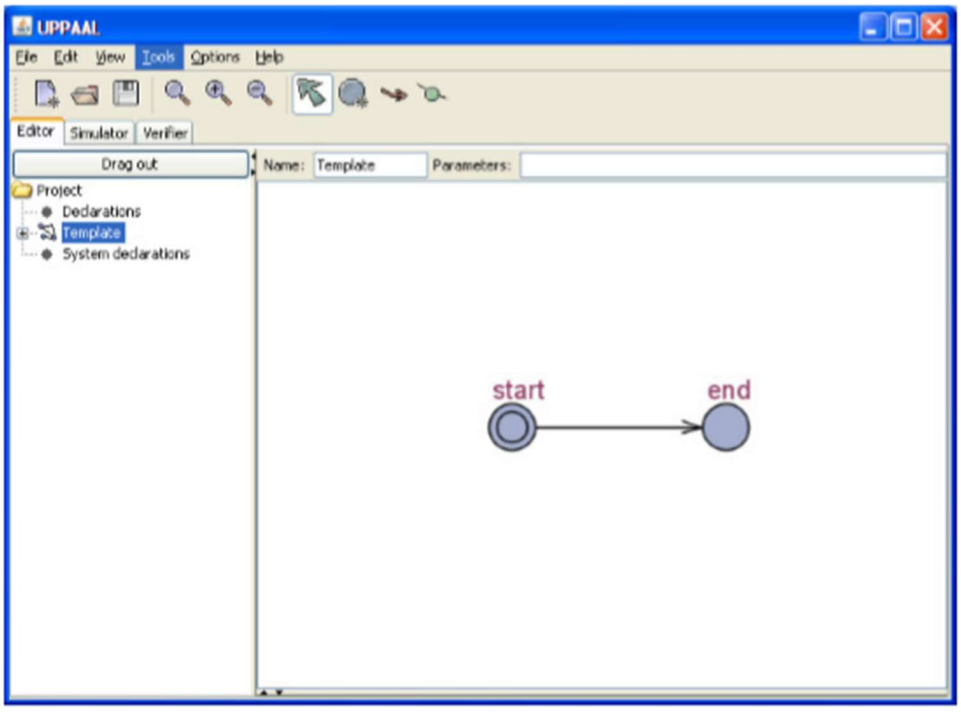


*Figure 22. Interface d'édition UPPAAL*

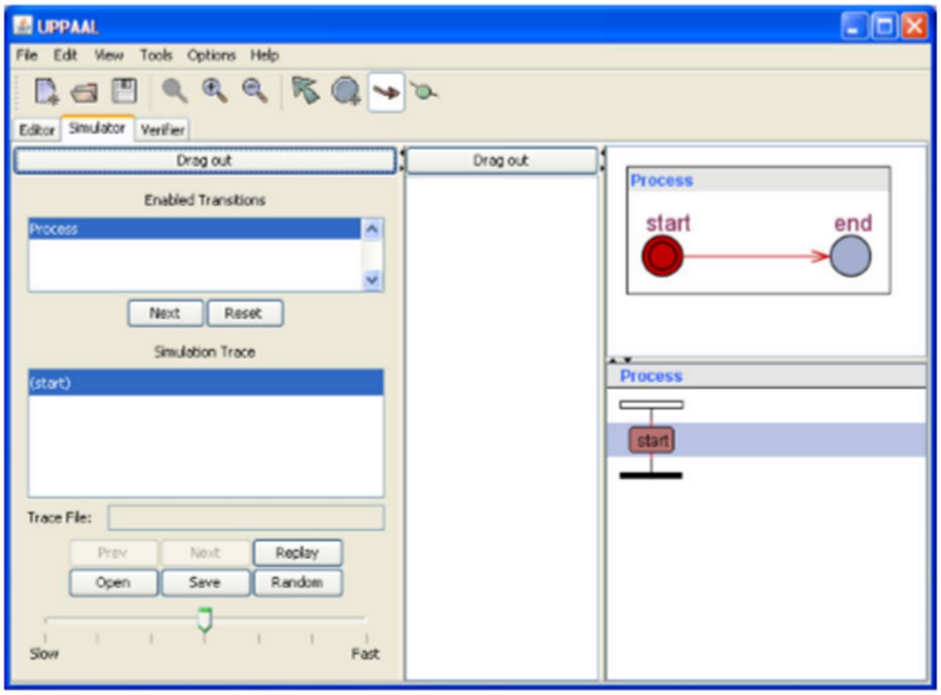


*Figure 23. Interface de simulation UPPAAL*

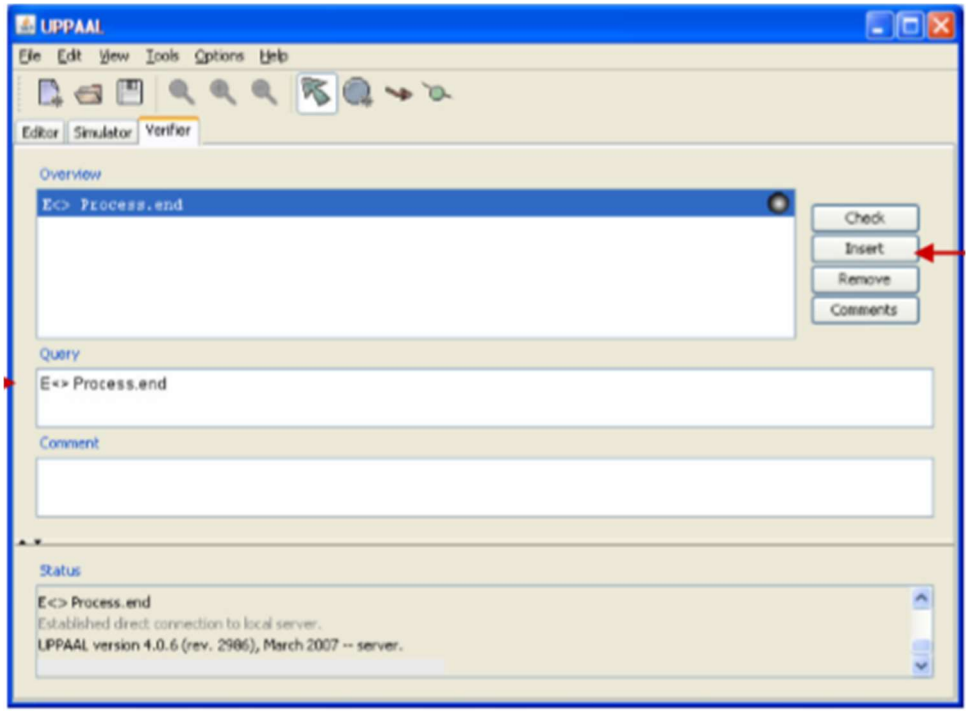


*Figure 24. Interface de vérification UPPAAL*

# Webographie


[1] https://www.artefacto-ar.com/realite-virtuelle/

[2] https://www.facilogi.com/blog/realite-virtuelle-dans-limmobilier/

[3] https://www.realite-virtuelle.com/chirurgiens-realite-virtuelle-entrainement/

[4] https://www.larousse.fr/dictionnaires/francais/sp%C3%A9cification/74085

[5] https://www.techno-science.net/definition/11508.html

[6] http://gpp.oiq.qc.ca/analyse_des_besoins.htm#:~:text=Analyse%20des%20besoins,%C3%A9pondant%20aux%20besoins%20du%20client.

[7] https://www.irif.fr/~sighirea/cours/genielog/c4-spec-adt.pdf

[8] https://www.memoireonline.com/07/08/1363/m_conception-realisation-application-gestion-centre-kinesie9.html#:~:text=I.,comprendre%20le%20contexte%20du%20syst%C3%A8me.

[9] https://cours-examens.org/images/Etudes_superieures/Ingenieur-en-informatique/2_annee/Intro_syst_info/cours_UQAC/14_La%20specification_des_besoins.pdf

[10] https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-cas-utilisation

[11] https://en.wikipedia.org/wiki/Verification_and_validation

[12] https://www.cloudnetcare.fr/les-3-typologies-de-dysfonctionnement-sur-vos-logiciels-et-applicatifs

[13] https://en.wikipedia.org/wiki/Verification_and_validation

[14] https://dept-info.labri.fr/~felix/Annee2008-09/S4/McInfo_ASR%20Tests/4.pdf

[15] https://www.irif.fr/~sighirea/cours/genielog/c11-12-vv.pdf

[16] https://www.embedded.com/an-introduction-to-model-checking/

[17] http://deptinfo.unice.fr/twiki/pub/Minfo05/BenoitPascal/modelChecking.pdf

[18] https://labo.fnac.com/actualite/hp-reverb-g2-casque-vr-valve-microsoft/

# Bibliographie


[19] Hocine, Mokrani & Ameur-Boulifa, Rabéa & Coudert, Sophie & Encrenaz, Emmanuelle. (2011). Approche pour l'intégration du raffinement formel dans le processus de conception des SOC. Journal Européen des Systèmes Automatisés. 45. 221-236. 10.3166/jesa.45.221-236.

[20] Rehfeld, *Estimating latency and concurrency of asynchronous real-time interactive systems using model checking, 2016 IEEE Virtual Reality (VR), Greenville, SC, 201( pp. 57-66, doi: 10.1109/VR.2016.7504688*

[21] R. Burch, *"Sequential circuit verification using symbolic model checking," 27th ACM/IEEE Design Automation Conference, Orlando, FL, USA, 1990, pp. 46-51, doi 10.1109/DAC.1990.114827.*

[22] Sekou Kangoye. Elaboration d'une approche de vérification et de validation de logiciel embarqué automobile, basée sur la génération automatique de cas de test. Systèmes embarqués. Université d'Angers, 2016. Français. ffNNT : 2016ANGE0017ff. fftel-01433015f

[23] L. J. Morell, *A Theory of Error-based Testing,* University of Maryland at College Park, 1984

[]24 Aït-Ameur Y., *Using the B formal approach for incremental specification design of interactive systems.,* 1999.

[25] André A. Geraldes, *Formal Verification of Medical CPS: A Laser Incision Case Study. ACM Trans. Cyber-Phys. Syst. 2, 4, Article 35,* juillet 2018.

[26] *Mass-spring Models for Physics-based,* 2018.

[27] Jarillo-Silva, *PHANToM OMNI Haptic Device: Kinematic and Manipulability.10.1109/CERMA.2009.55.*

[28] Vincent Chapurlat. Vérification et validation de modèles de systèmes complexes: application à la Modélisation d'Entreprise. Sciences de l'ingénieur [physics]. Université Montpellier II - Sciences et Techniques du Languedoc, 2007. ⟨tel-00204981⟩

[29] Amira Methni. Méthode de conception de logiciel système critique couplée à une démarche de vérification formelle. Logique en informatique [cs.LO]. Conservatoire national des arts et metiers - CNAM, 2016. Français. ⟨NNT : 2016CNAM1057⟩. ⟨tel 01445983⟩

[30] Rebaiaia, Mohamed-Larbi. Spécification et Vérification des Systèmes Critiques : Extension de l'Environnement VALID pour la Prise en Charge du Temps Réel. (2005).