République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Mémoire de Master

Présenté à l'Université Echahid Echeikh Larbi Tébessi - Tébessa
Faculté des Sciences Exactes et Sciences de la Nature et de la Vie

Département de : **Mathématiques et Informatique**
Domaine : **Informatique**
Spécialité : **Informatique**
Option : **SI**

Par : **Mr. DEBAILIA Achraf**

Développement d'un outil pour la génération du profil opérationnel pour les systèmes multi-agents normatifs

JURY

Président	M.A.A	**HAMIDANE Fathi**	Université Echahid Echeikh Larbi Tébessi - Tébessa
Rapporteur	M.A.A	**MENASSEL Yahia**	Université Echahid Echeikh Larbi Tébessi - Tébessa
Examinateur	M.A.B	**DAOUADI Khaireddine**	Université Echahid Echeikh Larbi Tébessi - Tébessa

2022/2023

Dédicaces

بسم الله الرحمن الرحيم

Je dédie cet humble travail

A mes chers parents.

Pour leur patience, leur amour, leur soutien et leur encouragement.

A mes chers frères et sœurs

A ma famille...

A mes amis et toute personne que j'aime...

Debailia Achraf

Résumé

Les systèmes multi-agents (SMA) normatifs sont basés sur les concepts d'institutions et de normes, régissant l'organisation des agents, leur comportement et leurs interactions. Bien qu'ils aient apportés des éléments de réponses très importants à plusieurs questions et problèmes soulevés par la communauté des chercheurs dans le domaine, leur développement n'est pas encore mature et il se fait de façon ad hoc et pragmatique. Ces lacunes affectent directement la qualité tant au niveau processus de développement que produit logiciel.

Dans cette étude, nous proposons une nouvelle approche pour tester la fiabilité des SMA normatifs en se basant sur méthodologie développée récemment [76]. Le présent travail, qui se place dans le contexte du génie logiciel orienté agent, vise aussi le développement d'un outil pour la génération automatique du profil opérationnel (PO) pour les SMA normatifs.

Mots-clés : SMA normatifs, Profil Opérationnel (PO), Test, Fiabilité.

Abstract

Normative multi-agent systems (MAS) are based on the concepts of institutions and norms, governing the organization of agents, their behavior and their interactions. Although they have provided important answers to many of the questions and problems raised by the research community in this field, their development is not yet mature, and is carried out in an ad hoc and pragmatic way. These shortcomings have a direct impact on the quality of both the development process and the software product.

In this study, we propose a new approach to testing the reliability of normative MAS, based on a methodology developed recently [76]. The present work, which takes place in the context of agent-oriented software engineering, also aims to develop a tool for the automatic generation of the operational profile (OP) for normative MAS.

Keywords: Normative MAS, Operational Profile (OP), Test, Reliability.

ملخص

تعتمد الأنظمة المعيارية متعددة الوكلاء (MAS) على مفاهيم المؤسسات والقواعد التي تحكم تنظيم الوكلاء وسلوكهم وتفاعلاتهم. على الرغم من أنهم قدموا إجابات مهمة للغاية على العديد من الأسئلة والمشكلات التي أثارها مجتمع الباحثين في هذا المجال، إلا أن تطورهم لم ينضج بعد ويتم ذلك بطريقة مخصصة وعملية. تؤثر أوجه القصور هذه بشكل مباشر على الجودة في كل من عملية التطوير ومستوى منتج البرنامج.
في هذه الدراسة، نقترح نهجًا جديدًا لاختبار موثوقية الأنظمة المعيارية متعددة الوكلاء استنادًا إلى منهجية تم تطويرها مؤخرًا [76]. يهدف هذا العمل، الذي تم وضعه في سياق هندسة البرمجيات الموجهة للوكيل، أيضًا إلى تطوير برنامج لإنشاء للملف التشغيلي (OP) للأنظمة المعيارية متعددة الوكلاء تلقائيا.

كلمات مفتاحية: الأنظمة متعددة الوكلاء المعيارية، الملف التشغيلي (OP)، الإختبار، الموثوقية.

Remerciements

قال رسول الله صلى الله عليه وسلم:

"من لم يشكر الناس لم يشكر الله ومن اهدى اليكم معروفا فكافئوه فان لم تستطيعوا فادعوا له."

C'est avec un grand plaisir que je réserve ces mots en signe de reconnaissance à ceux qui ont contribué de près ou de loin à l'élaboration de ce modeste travail.

Tout d'abord, je tiens à remercier Allah tout puissant, de m'avoir permis de mener à bien ce Mémoire, et de m'avoir orienté au chemin du savoir.

Ensuite, je voudrais exprimer mes vifs remerciements :

- *À Mr.* ***MENASSEL Yahia****, maitre-assistant au sein du département mathématiques et informatique de la faculté des sciences exactes et sciences de la nature et de la vie à l'université echahid echaikh larbi tébessi, pour la confiance qu'il m'a témoignée en acceptant de m'encadrer dans ce travail, et qui n'a cessé de me prodiguer ses conseils et ses suggestions pertinentes. Je lui en serais gré.*
- *Au président de jury, Mr.* ***HAMIDANE Fathi****, maitre-assistant au sein du département mathématiques et informatique de la faculté des sciences exactes et sciences de la nature et de la vie à l'université echahid echaikh larbi tébessi, qui a bien voulu accepter de me faire un grand honneur pour présider le jury de ma soutenance.*
- *À l'examinateur, Mr.* ***DAOUADI Kheireddine****, maitre-assistant au sein du département mathématiques et informatique de la faculté des sciences exactes et sciences de la nature et de la vie à l'université echahid echaikh larbi tébessi, qui a aimablement accepté d'apporter ses compétences et de siéger parmi le jury de ce mémoire et d'avoir fait partie de la commission de lecture.*

Enfin, j'adresse mes chaleureux remerciements à ma famille et mes amis pour leur soutien. Je les remercie d'avoir toujours cru en moi, cela m'a permis d'atteindre mes objectifs. Puissent-ils trouver ici l'expression de ma gratitude.

Tous vos efforts ont fructifié.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

" Les sciences n'existent et ne progressent que par le conflit des idées et théories ainsi que par la difficile victoire d'une théorie déviante sur les idées dominantes et reçues."

Edgar Morin

1. Contexte et problématique

Il est malheureusement fréquent de constater des erreurs lors du développement de dispositifs programmés. Ces erreurs doivent être prises au sérieux car notre vie quotidienne dépend de plus en plus des systèmes logiciels, et des pertes économiques importantes, voire des conséquences fatales, peuvent en découler. Il est donc essentiel de valider les systèmes programmés avant leur mise en service afin d'atteindre un niveau de confiance satisfaisant. Pour cela, les opérateurs économiques, les académiciens et les chercheurs doivent développer et améliorer des méthodes et des outils appropriés pour la validation des systèmes logiciels. Il y a quelques décennies, un logiciel ne comportait que quelques milliers de lignes de code, mais aujourd'hui, ces produits sont couramment constitués de plusieurs millions de lignes. Cette complexité accrue entraîne naturellement une augmentation des erreurs lors du développement, notamment lors de la programmation.

Le test de logiciel est une étape cruciale du développement de logiciels, visant à améliorer la qualité du produit en identifiant et en corrigeant les erreurs et les anomalies. Il s'agit d'une activité qui implique l'exécution d'un système dans des conditions spécifiques, afin d'observer et d'enregistrer les résultats obtenus et de les comparer aux caractéristiques ou résultats prévus.

Les systèmes multi-agents (SMA) sont des systèmes complexes. Par conséquence, il est difficile de les corriger et les tester. La tâche de tester les SMA est complexe en raison de leur nature distribuée, autonome et délibérative. Ces caractéristiques uniques des agents logiciels rendent l'utilisation des méthodes de test actuelles difficile.

Dans la littérature, il existe plusieurs approches et techniques proposées pour le but de tester les SMA. Parmi ces techniques, il n'y a aucune qui assure la fiabilité de ce genre de systèmes. Ce mémoire s'intéresse à une nouvelle approche de test de fiabilité des SMA normatifs.

2. Objectif

Notre travail consiste à :

- Proposer une nouvelle approche de test de fiabilité des SMA normatifs.
- Développer un outil pour la génération automatique du profil opérationnel pour les SMA normatifs.

3. Organisation du manuscrit

La suite de ce manuscrit est structurée comme suit :

- *Chapitre 1. Les systèmes multi-agent normatifs.* Dans ce chapitre, une vue générale sur les systèmes multi-agents normatifs sera présentée.
- *Chapitre 2. Etude du domaine : Langages et outils de transformation.* Ce chapitre présente un état de l'art, d'une part, sur le test des logiciels et SMA, et d'autre part, sur le test de fiabilité des logiciels.
- *Chapitre 3. Vers une nouvelle approche de test de fiabilité des systèmes multi-agents normatifs.* Ce chapitre est composé de deux parties essentielles : La première consiste à proposer une nouvelle approche afin de tester la fiabilité des SMA normatifs quant à la deuxième partie est consacrée pour le développement d'un outil pour la génération automatique du profil opérationnel pour les SMA normatifs.
- Nous conclurons ce manuscrit par une conclusion générale qui discute les apports de notre travail ainsi que les perspectives envisagées en vue d'ouvrir de nouvelles directions.

Chapitre 1

Les systèmes multi-agents normatifs

" Tout grand progrès scientifique est né d'une nouvelle audace de l'imagination."

John Dewey

Introduction

Les systèmes multi-agents sont un paradigme de programmation récent qui s'appuie sur l'intelligence artificielle distribuée (DAI). Ils se composent d'un ensemble d'agents autonomes qui interagissent entre eux. Ce modèle est largement considéré comme l'un des plus adaptés pour la création de systèmes complexes, tels que la simulation de phénomènes sociaux et biologiques. De nombreuses études ont proposé l'introduction de normes ou de lois pour réguler le comportement des agents, ce qui a donné naissance aux systèmes multi-agents normatifs (Nor-MAS). Dans ce chapitre, nous présenterons les systèmes multi-agents normatifs. Nous allons commencer par une introduction aux systèmes multi-agents, et par la suite, nous présenterons les concepts fondamentaux de ce type de systèmes (Nor-MAS) [1].

1. Les systèmes multi-agents

Le domaine des systèmes multi-agents (SMA) est un domaine de recherche en constante évolution qui sert d'interface entre plusieurs disciplines, notamment l'intelligence artificielle, les systèmes distribués et le génie logiciel. Il se concentre sur l'étude des comportements collectifs qui résultent des interactions entre des entités autonomes et flexibles, appelées agents, dans un environnement donné. Les interactions entre agents peuvent prendre la forme de coopération, de compétition ou de simultanéité. Selon la méthode des voyelles proposée par *Demazeau* en 1996, un système multi-agents est constitué de quatre éléments de base : l'agent (A), l'environnement (E), l'organisation (O) et l'interaction (I). Dans les sections suivantes, nous examinerons ces concepts fondamentaux en détail [2].

1.1. Agent

La signification du terme "agent" demeure encore controversée, malgré les efforts croissants de standardisation des aspects opérationnels liés à la conception des systèmes multi-agents (SMA) qui sont de plus en plus reconnus. Selon Ferber [3], un agent est défini comme une entité active qui peut :

- Réagir dans un environnement.
- Faire des interactions dans l'environnement et avec d'autres agents.
- Percevoir son environnement (avec une représentation partielle).
- Poursuit un objectif individuel.
- Possède des compétences et peut éventuellement se reproduire.

D'autre part, *Wooldridge* [4] a introduit une définition alternative du concept agent. Selon ce dernier, l'agent est « un programme informatique qui est situé dans un environnement et qui est doté de comportements autonomes (actions) lui permettant d'atteindre, dans cet environnement, les objectifs qui lui ont été fixé à sa conception ».

La communauté de recherche parvient presque à un consensus sur les caractéristiques des agents, en dépit de la diversité des définitions qui existent. En effet, selon [4] et [5], un agent est généralement caractérisé par :

- *Autonomie* : est un concept fondamental dans la définition du comportement des agents, qui leur permet d'agir sans intervention extérieure et de contrôler leurs propres actions.
- *Réactivité* : est la capacité de l'agent à percevoir son environnement et à y répondre rapidement.
- *Proactivité* : est la capacité de l'agent de prendre l'initiative en démontrant des comportements orientés objectifs.
- *Mobilité* : est la capacité de l'agent de se déplacer dans un réseau.
- *Véracité* : l'agent ne communique pas de fausses informations.
- *Bienveillance* : implique l'absence de buts contradictoires.
- *Rationalité* : l'agent peut atteindre leurs objectifs avec le moins de coûts possible.

Il est important de noter que ces caractéristiques ne sont pas obligatoires pour tous les agents, car leur présence dépend fortement du type d'agent. De plus, la présence de certaines caractéristiques peut conduire à l'émergence d'un type spécifique d'agent, tandis que dans d'autres cas, une caractéristique peut servir de critère de classification des agents. Parmi ces critères, on peut identifier la granularité comme une caractéristique de classification primordiale pour les agents. La granularité représente la complexité des agents et de leurs comportements. Selon cette caractéristique, on distingue trois types d'agents [3] [6] :

- ***Agents réactifs*** : Ils sont des outils qui peuvent être exploités pour remplir une fonction spécifique, comme répondre à un besoin interne ou atteindre un objectif prédéfini par le créateur. Ils ne disposent d'aucune aptitude à la réflexion, mais sont plutôt caractérisés par des actions simples qui sont entièrement guidées par les conditions actuelles de leur environnement. Ces comportements sont souvent représentés par des modèles de machines à états finis.
- ***Agents cognitifs*** : Ce type des agents, grâce à leur complexité et à leurs capacités de raisonnement, est capable de fonctionner de manière autonome. Ils sont confrontés à des tâches plus complexes que celles auxquelles sont confrontés les agents réactifs, et sont également en mesure de résoudre des problèmes plus complexes. Leurs mécanismes internes de représentation et de raisonnement leur permettent de travailler indépendamment des autres agents, offrant ainsi une grande flexibilité dans leur comportement.
- ***Agents hybrides*** : Dans ce type d'agent, chaque module gère de manière autonome la partie réflexe (réactive) et raisonnée (cognitive) du comportement de l'agent.

1.2. Environnement

En général, l'environnement [1] est le cadre dans lequel les agents évoluent. Il s'agit d'une des briques fondamentales et très importantes dans les systèmes multi-agents (SMA), car il offre un moyen de

communication entre les agents. Chaque environnement comprend un ensemble d'objets passifs tels que des ressources, des capteurs, des obstacles, etc., qui peuvent être manipulés par les agents.

Selon *Weyns*, *Omicini* et *Odell* [7], l'environnement est considéré comme une abstraction de premier niveau dans les systèmes multi-agents (SMA) et remplit un double rôle. Tout d'abord, il fournit les conditions environnementales dans lesquelles l'agent existe, soulignant ainsi son importance au sein de chaque SMA. De plus, il offre une abstraction de conception réalisable pour la construction d'applications SMA. Par conséquent, l'environnement devient une composante intégrante du système multi-agents (MAS) et les ingénieurs peuvent l'utiliser de manière créative lors de la conception de SMA.

Chaque environnement est représenté par un ensemble des propriétés. *Russell* et *Norvig* [8] proposent la classification suivante des propriétés de l'environnement :

- *Accessible/Inaccessible* : Un environnement accessible est un environnement qui permet à l'agent d'obtenir des informations précises, complètes et à jour sur son état.
- *Déterministe/Non déterministe* : Un environnement est considéré comme déterministe lorsqu'il est possible de prévoir avec certitude l'effet de chaque action entreprise. Ainsi, il n'y a aucune incertitude quant à l'état qui résultera de l'exécution d'une action donnée.
- *Statique/Dynamique* : On peut considérer un environnement comme statique lorsqu'il demeure inchangé, sauf en cas d'actions entreprises par l'agent. En revanche, un environnement dynamique est soumis à l'influence d'autres processus qui le modifient indépendamment de la volonté de l'agent. Le monde physique et Internet sont des exemples d'environnements hautement dynamiques.
- *Discrète/continue* : Un environnement est considéré comme discret lorsqu'il renferme un nombre limité et défini d'actions et de perceptions.

Généralement, le type d'environnement le plus complexe est celui qui est inaccessible, non déterministe, dynamique et continue.

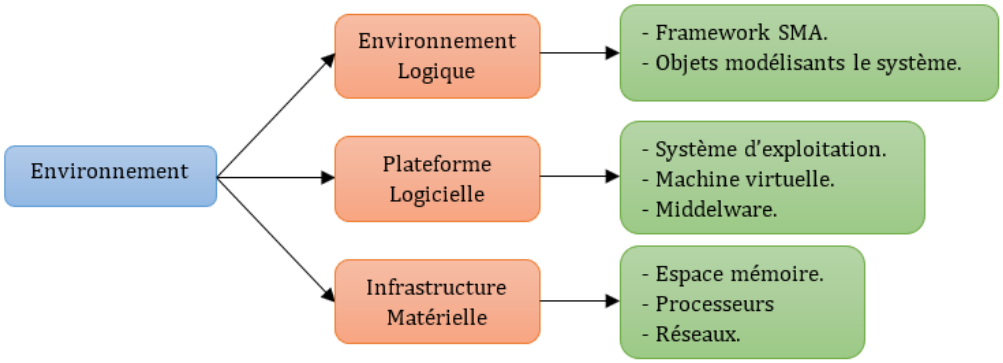

Figure 1. Le modèle en couches d'un environnement.

Pour éviter toute confusion quant au rôle et à l'architecture d'un environnement dans les systèmes multi-agents (SMA), une architecture en couches de l'environnement est proposée. Cette architecture se compose de trois couches distinctes :

- La *couche matérielle*, qui englobe l'infrastructure matérielle nécessaire au fonctionnement de l'environnement. Cette couche définit les composants physiques tels que les ordinateurs, les capteurs, les réseaux, etc.

- Au-dessus, nous avons la *couche logicielle*, qui comprend des systèmes tels que les systèmes d'exploitation et les bibliothèques logicielles. Elle fournit les fonctionnalités et les services nécessaires au bon fonctionnement de l'environnement.
- Enfin, au sommet de l'architecture en couches, nous trouvons la *couche logique*, qui représente la plateforme de développement pour les SMA. Cette couche offre les outils et les frameworks permettant aux ingénieurs de concevoir, développer et déployer des systèmes multi-agents.

La figure 1 représente le modèle de cette architecture, mettant en évidence la hiérarchie des couches et leur interrelation.

Tableau 1. Classification des situations d'interactions [3].

Buts	Ressources	Compétences	Type de situation	Remarque
Compatibles	Suffisantes	Suffisantes	Indépendance	Situation d'indifférence
Compatibles	Suffisantes	Insuffisantes	Collaboration	Simples situations de coopérations
Compatibles	Insuffisantes	Suffisantes	Encombrement	
Compatibles	Insuffisantes	Insuffisantes	Coordonnée	
Incompatibles	Suffisantes	Suffisantes	Compétition individuelle pure	Situations d'antagonismes
Incompatibles	Suffisantes	Insuffisantes	Collective pure	
Incompatibles	Insuffisantes	Suffisantes	Conflits individuels pour des ressources	
Incompatibles	Insuffisantes	Insuffisantes	Conflits collectifs pour des ressources	

1.3. Interaction

Selon *Ferber* [3], l'interaction dans un système multi-agents (SMA) se réfère à la mise en relation dynamique de deux agents ou plus par le biais d'une série d'actions réciproques. L'interaction se manifeste par des actions qui ont des conséquences sur le comportement futur des agents, créant ainsi une séquence d'interactions.

Les agents interagissent en utilisant la communication comme moyen. On peut distinguer deux types de communication : la communication indirecte par le biais de l'environnement et la communication directe par l'échange de messages entre les agents. Dans la communication indirecte, les agents communiquent en utilisant l'environnement comme médiateur, en déposant des informations ou en agissant sur des objets dans l'environnement pour transmettre des messages aux autres agents. En revanche, dans la communication directe, les agents échangent des messages entre eux en utilisant un langage de communication spécifique tel que KQML ou FIPA ACL.

Ferber a proposé une classification exhaustive des situations d'interactions selon plusieurs critères: la *compatibilité des buts*, la *suffisance des ressources* et la *suffisance des compétences*. Ces situations allant de simple situation d'indifférence jusqu'aux des situations de conflits collectifs pour des ressources. Le tableau 1 résume ces situations.

1.4. Organisation

Lors de la conception d'un système multi-agents, il est possible de voir à travers deux angles différents [2]:

- Aspect micro (orienté agent) : selon ce point de vue, l'accent est mis sur la façon dont nous concevons et construisons un agent autonome pour qu'il puisse agir de manière indépendante.
- Aspect macro (orienté organisation) : ce concept est basé sur la façon dont nous obtenons une société des agents coopérant efficacement.

Un système multi-agents est essentiellement une société organisée, où l'organisation joue un rôle central. Une organisation représente un arrangement de relations entre des composants ou des individus qui génère une unité ou un système avec des caractéristiques qui ne sont pas présentes au niveau des composants ou des individus individuellement. Les organisations établissent des liens entre divers éléments, événements ou individus, contribuant ainsi à former un ensemble cohérent. Elles assurent une solidarité et une stabilité relative, permettant ainsi au système de perdurer malgré les perturbations aléatoires [5].

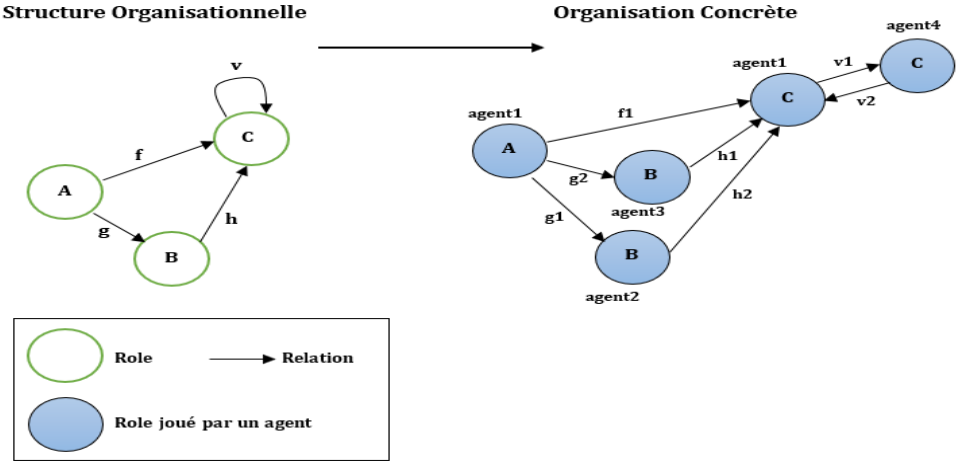

Figure 2. Structure organisationnelle et Organisation concrète [1].

De manière simplifiée, une organisation est définie par un ensemble de rôles joués par les agents. La structure organisationnelle consiste en l'ensemble des rôles et des relations entre eux. En assignant des rôles spécifiques aux différents agents du système, une organisation concrète est établie, comme illustré dans la figure 2.

Pour étudier les organisations multi-agents, plusieurs modèles ont été proposés pour simplifier les relations entre les concepts organisationnels, tels que AGR (Agent/Group/Role) et MOISE+ (Model of Organized Information and Social Environment). Ces modèles offrent des cadres pour décrire et analyser les aspects organisationnels des systèmes multi-agents.

1.5. Domaines d'application des système multi-agents

Les systèmes multi-agents trouvent une large gamme d'applications dans divers domaines. D'après Ferber, il existe cinq grandes catégories d'applications pour les systèmes multi-agents [3] :

- Résolution de problèmes généralisés : Les systèmes multi-agents peuvent être utilisés pour résoudre des problèmes complexes et difficiles à résoudre de manière traditionnelle. Les agents interagissent et collaborent pour trouver des solutions efficaces.
- Robotique distribuée : Les systèmes multi-agents sont utilisés pour la coordination et le contrôle de robots distribués, leur permettant de travailler ensemble de manière autonome pour accomplir des tâches complexes.

- Simulation multi-agents : Les systèmes multi-agents sont largement utilisés dans les simulations pour modéliser des phénomènes sociaux, économiques, écologiques, etc. Les agents représentent des entités individuelles qui interagissent pour reproduire les comportements observés dans le monde réel.
- Construction de mondes hypothétiques : Les systèmes multi-agents peuvent être utilisés pour créer des environnements virtuels où des agents autonomes évoluent et interagissent. Ces mondes hypothétiques sont utilisés pour la recherche, la formation, les jeux, etc.
- Conception kénétique de programmes : Les systèmes multi-agents peuvent être utilisés pour concevoir des programmes informatiques complexes de manière évolutive et auto-adaptative. Les agents agissent comme des unités de traitement indépendantes qui coopèrent pour atteindre des objectifs spécifiques.

La figure 3 présente une synthèse visuelle de ces domaines d'application des systèmes multi-agents.

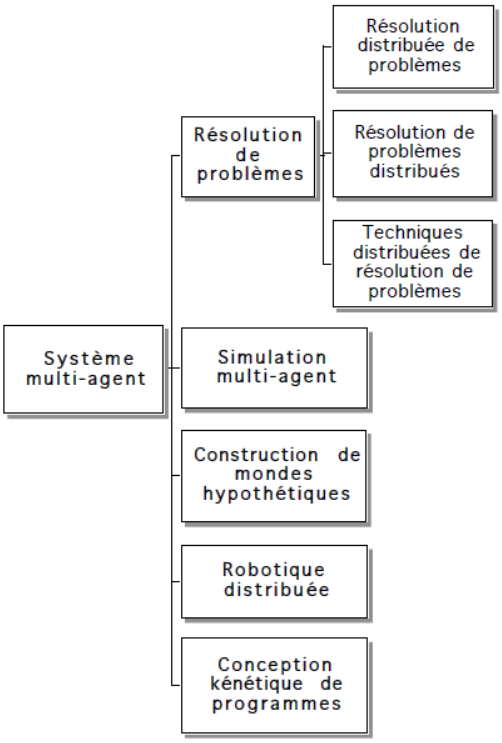

Figure 3. Le domaine d'application des système multi-agents [3].

2. Les systèmes normatifs

La définition du terme "norme" peut varier selon le domaine d'application. Par exemple, le dictionnaire Larousse propose différentes définitions pour ce mot en fonction du contexte spécifique. Dans le domaine industriel, une norme est une règle établissant les conditions d'exécution d'une action, la réalisation d'un objet ou le développement d'un produit, dans le but d'assurer l'uniformité ou l'interchangeabilité [9].

En linguistique, une norme désigne un système d'instructions définissant les choix à faire parmi les usages d'une langue pour se conformer à un idéal esthétique ou socioculturel spécifique.

En mathématiques, une norme est un ensemble de règles qui vérifient certaines conditions particulières, souvent liées à des mesures de distance ou de taille.

En psychologie, une norme est un étalonnage d'un test, c'est-à-dire une référence permettant de comparer les performances individuelles à celles d'un groupe représentatif.

Ces différentes définitions illustrent la diversité des domaines où le concept de norme est appliqué, chacun ayant sa propre signification et ses propres implications.

D'une manière plus simple, une norme est un principe d'action juste qui s'impose aux membres d'un groupe et sert à guider, contrôler ou réglementer un comportement approprié et acceptable dans le système normatif afin de réduire les conflits et améliorer la coordination [10]. Il est effectivement possible de distinguer deux grandes catégories de normes : les *normes conventionnelles* et les *normes essentielles*.
Les normes conventionnelles sont des normes qui émergent naturellement sans qu'il y ait une application intentionnelle. Elles se développent souvent à travers des pratiques sociales et des comportements observés au sein d'une communauté ou d'une société. Les normes conventionnelles peuvent être liées à des coutumes, des traditions, des habitudes ou des conventions établies. Elles peuvent varier d'une culture à une autre et peuvent évoluer au fil du temps.
Quant aux normes essentielles, elles sont conçues pour favoriser le travail collectif et résoudre les situations conflictuelles entre les intérêts individuels et collectifs. Ces normes sont généralement établies dans le but de promouvoir la coopération, la coordination et l'harmonie entre les membres d'un groupe ou d'une organisation. Elles peuvent être formelles ou informelles, et peuvent être soutenues par des institutions, des règlements ou des accords spécifiques [11].
En résumé, les normes conventionnelles émergent naturellement tandis que les normes essentielles sont établies dans le but de promouvoir le travail collectif et de résoudre les conflits entre les intérêts individuels et collectifs.

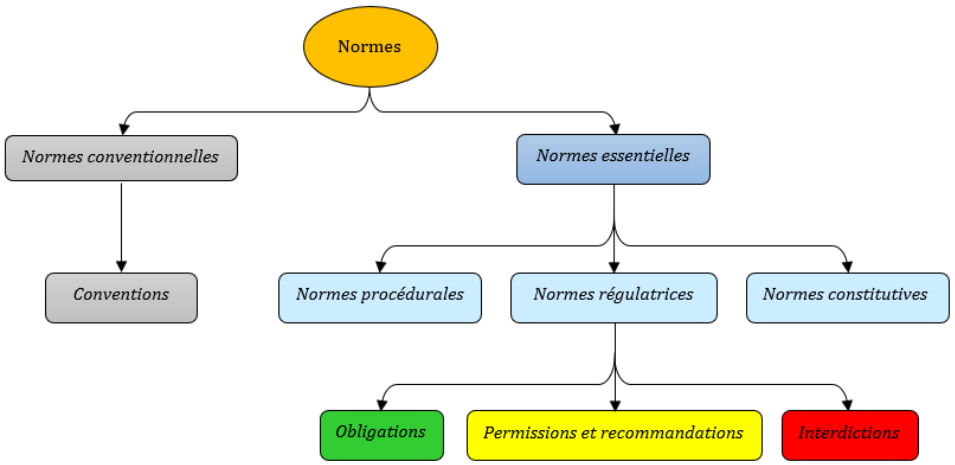

Figure 4. Classification des normes.

Selon *Boella, van der Torre, & Verhagen* [12], il existe trois types de normes :

- **Normes constitutives** : Elles sont employées pour créer de nouvelles normes et pour déterminer l'appartenance d'un agent à un système d'action donné.
- **Normes procédurales** : Elles sont des normes importantes pour les agents impliqués dans le système normatif visant à maintenir l'ordre social, en particulier en respectant des critères objectifs.
- **Normes régulatrices** : Elles définissent les comportements optimaux ainsi que les différents degrés de comportements non optimaux du système. Elles sont exprimées en termes d'obligations, d'interdictions et d'autorisations [11].

Leur objectif est de réguler les activités en imposant des obligations ou des interdictions. On peut distinguer plusieurs types d'action :

- *Obligation* : dans ces normes, les comportements doivent être respectés. Tout manquement à leur application est considéré comme une violation de la norme, ce qui entraîne des conséquences.
- *Recommandation* : L'exécution d'un comportement n'est pas obligatoire, mais si ce comportement est exécuté, cela entraîne une récompense.
- *Interdiction* : c'est l'inverse de l'obligation. Un comportement particulier entraîne une sanction en raison de la violation de la norme.
- *Permission* : dans ce cas, l'exécution d'un comportement spécifique n'a aucune conséquence.

3. Les systèmes multi-agents normatifs (Nor-MAS)

Un système multi-agents normatif est une composition de deux domaines : les systèmes multi-agents et les systèmes normatifs. En fait, les concepts normatifs sont utilisés pour faciliter la coopération et la coordination entre les groupes sociaux d'un système multi-agents [11].

Boella et *van der Torre* définissent un système multi-agent normatif comme «un système multi-agent organisé par des mécanismes pour représenter, communiquer, distribuer, détecter, créer, modifier et appliquer des normes, et délibérer sur les normes et détecter la violation et l'accomplissement des normes» [13].

Effectivement, un système multi-agents normatif peut être défini comme un système multi-agents dans lequel des systèmes normatifs sont intégrés. Dans ce contexte, les agents ont la capacité de décider s'ils souhaitent suivre ou ne pas suivre les normes qui sont explicitement représentées. Les systèmes normatifs spécifient également comment et dans quelle mesure les agents peuvent modifier ces normes [12]. Cette approche normative permet de modéliser et de réguler les comportements des agents au sein du système. Les normes jouent un rôle essentiel dans la coordination, la coopération et le contrôle des interactions entre les agents. Elles fournissent un cadre de référence pour guider les actions des agents et favoriser le respect des règles et des attentes au sein du système multi-agents.

En intégrant des systèmes normatifs, les systèmes multi-agents normatifs offrent la possibilité d'explorer des questions telles que la conformité, la déviance, la sanction et l'évolution des normes au sein du système. Cela permet d'étudier et de comprendre les dynamiques sociales complexes qui émergent de l'interaction entre les agents et les normes au sein de l'environnement multi-agent.

En se basant sur les définitions précédentes, on peut conclure que la représentation des normes est une question cruciale dans un système multi-agents normatif. La section suivante est consacrée à la présentation de cet axe.

3.1. Représentation des normes

Dans la représentation des normes, différentes méthodes ont été proposées dans la littérature spécialisée. Selon *Hollander* et *Annie* [14], les normes peuvent être formalisées selon quatre classes principales :

- *La logique déontique* : Cette approche repose sur l'utilisation de la logique déontique pour représenter les normes sous forme d'obligations, d'interdictions et de permissions. Elle permet de définir de manière précise ce qui est requis, ce qui est interdit et ce qui est autorisé dans un contexte donné.

- *Les systèmes à base de règles* : Cette approche consiste à recourir à des règles spécifiques pour représenter les normes, lesquelles décrivent les conditions et les actions à prendre en conséquence. Les systèmes basés sur des règles sont flexibles pour représenter des normes complexes avec des conditions multiples.
- *Les chaînes binaires* : Cette approche utilise des séquences de 0 et de 1 pour représenter les règles. Chaque chiffre binaire est associé à une condition particulière qui indique si une règle est respectée ou non. Bien que cette représentation binaire soit simple et concise, elle peut être limitée en termes d'expressivité.
- *La théorie des jeux* : Cette approche s'appuie sur les principes et les schémas de la théorie des jeux pour décrire les normes. Les normes sont définies en termes de tactiques et de résultats anticipés dans des contextes d'interaction entre les individus. La théorie des jeux permet d'examiner les motivations et les attitudes des individus à l'égard des normes.

Ces différentes approches de représentation des normes fournissent des moyens diversifiés pour modéliser et analyser les interactions normatives dans les systèmes multi-agents. Chaque classe a ses propres avantages et inconvénients, et le choix de l'approche appropriée dépend du contexte et des objectifs de la modélisation.

3.2. Le mécanisme d'application des normes

La conception de mécanismes d'application est un enjeu crucial dans les systèmes multi-agents normatifs. En effet, les agents étant autonomes, ils peuvent ne pas respecter les comportements attendus, d'où l'importance de ces mécanismes pour les contraindre à se conformer aux normes. Ils ont pour rôle de surveiller les comportements des agents et de les sanctionner en cas de violation des normes.

Il est primordial de motiver les agents à respecter les normes en les incitant à les suivre ou à les abandonner. En effet, leur adhésion dépend de leurs objectifs et de leur autonomie. Toutefois, il est crucial de favoriser la conformité aux normes, que ce soit par des moyens internes tels que la honte, la culpabilité ou la réputation, ou par des moyens externes tels que les sanctions, les pénalités et les récompenses [11].

4. Simulation et modélisation des systèmes multi-agents

Les systèmes multi-agents offrent une approche novatrice pour la modélisation et la simulation en sciences de l'environnement. Ils permettent de représenter directement les individus, leurs comportements et leurs interactions, ce qui offre de nouvelles perspectives dans l'étude des systèmes complexes. Dans les simulations multi-agents, les individus sont modélisés sous forme d'agents, et le comportement global émerge de l'interaction entre ces agents. Par exemple, dans un modèle de population, chaque individu est représenté comme un agent et les interactions entre ces agents (coopération, lutte, reproduction) déterminent le nombre d'individus dans une espèce donnée.

L'objectif de ces simulations est de prendre en compte à la fois des paramètres quantitatifs (chiffres, nombres) et qualitatifs (comportements individuels avec des raisonnements stratégiques). Les modèles multi-agents adoptent une approche micro-analytique, en se concentrant sur le comportement des individus et leurs interactions avec l'environnement simulé [3].

Au fil du temps, de nombreux outils de modélisation et de simulation basés sur les agents ont été développés. Chaque outil a ses propres caractéristiques en termes de généralité, d'utilisabilité, de

modificabilité, d'extensibilité et de performance. Ces outils adoptent différentes stratégies de programmation et offrent des cadres conceptuels et des bibliothèques logicielles pour faciliter la conception et la description des modèles basés sur les agents. Les plates-formes de modélisation basées sur les agents les plus couramment utilisées sont souvent basées sur des frameworks et des bibliothèques. Elles proposent un ensemble de concepts standard pour la conception des modèles basés sur les agents et fournissent des outils de simulation. Ces frameworks facilitent la mise en œuvre des modèles et offrent une infrastructure pour la gestion des agents, les interactions et les comportements émergents [15].

En résumé, les outils de modélisation et de simulation basés sur les agents permettent d'explorer et d'étudier les systèmes complexes en prenant en compte les comportements individuels et les interactions entre les agents, ouvrant ainsi de nouvelles possibilités dans les sciences de l'environnement.

Les principales caractéristiques de la modélisation multi-agents sont leur capacité d'intégration et leur flexibilité. En fait, les variables quantitatives basées sur des règles symboliques, les équations différentielles et le comportement peuvent être intégrés dans la même modélisation. Les modifications d'intégration sont également faciles, chaque enrichissement du modèle est réalisé en ajoutant des règles de comportement qui fonctionnent au niveau individuel. De plus, les individus sont toujours différenciés les uns des autres, et de nouveaux types d'agents avec leurs propres modèles de comportement peuvent être ajoutés qui interagissent avec des agents déjà définis.

Les utilisateurs du simulateur multi-agents ont un rôle actif. Il utilise le système comme s'il s'agissait d'un laboratoire miniature, déplaçant les individus, modifiant leur comportement et modifiant les conditions environnementales. Chaque agent est clairement identifié comme un être naturel et peut suivre les individus à tout moment de leur évolution avec le niveau de compétence requis. Nous exploitons ensuite la puissance des ordinateurs pour traiter les données acquises, agrégées et exploitées, en utilisant des techniques statistiques pour tester les hypothèses proposées. Plusieurs simulateurs existent dans la littérature comme [3] [16] :

- **AgentSheets** : Est une Plateforme de programmation visuelle qui convient particulièrement aux débutants. Elle a été utilisée pour enseigner diverses matières telles que les mathématiques, les sciences et les études sociales. Contrairement à la programmation traditionnelle basée sur du texte, AgentSheets ne nécessite pas de codage, tout le développement se fait via une interface graphique intuitive.
- **Ascape** : Est un outil de développement et d'analyse de modèles basé sur les agents, créé par Miles Parker du Center for Social and Economic Dynamics de la Brookings Institution. Il est programmé en Java, ce qui nécessite des compétences en programmation Java et une compréhension de la philosophie orientée objet pour les développeurs qui souhaitent simuler des modèles dans Ascape.
- **SIMDELTA** : Est un simulateur qui a été développé par F. Bousquet et C. Cambier en 1995. Il permet de simuler la dynamique des populations de poissons en prenant en compte divers facteurs biologiques et topologiques qui peuvent influencer leur évolution, ainsi que les décisions des pêcheurs. Chaque pêcheur est représenté sous la forme d'un agent cognitif dont le comportement est décrit par un système à base de connaissances. Ce système est constitué d'une base de données contenant les croyances et les souvenirs de pêcheur, ainsi que d'une règle représentant les stratégies cognitives d'exploitation de la composition du système biome. Deux séries d'expériences ont été menées avec ce simulateur. La première étudie la dynamique des populations de poissons en fonction de l'augmentation de l'effort de

pêche, tandis que la seconde simule des pêcheurs prenant des décisions et agissant sur des réserves renouvelables. Ce modèle montre l'importance des mécanismes de prise de décision pour la dynamique des poissons.

- **SIMPOP** : Est une simulation qui permet de modéliser la dynamique de l'évolution des systèmes urbains sur une longue période de temps. Cette simulation prend en compte l'origine, le développement et la concentration des fonctions urbaines à différents niveaux. Pour cela, l'environnement est représenté par un ensemble de « lieux » de différentes tailles et formes, qui possèdent des ressources naturelles et des éléments tels que des voies de communication. Chaque lieu est représenté par une ville, qui est caractérisée par la taille de sa population, sa richesse économique et les fonctions qu'elle possède. Le comportement d'une ville est donc déterminé par la somme des comportements de ses habitants.

Conclusion

Ce premier chapitre résume des généralités sur les systèmes multi-agents (SMA), les systèmes normatifs ainsi que les systèmes multi-agents normatifs (Nor-MAS). Les SMA sont composés d'un ensemble d'agents en interaction. Cette interaction peut prendre plusieurs formes comme la coopération, la coordination et la résolution des conflits. Afin de mieux gérer les situations de coordination entre les agents au sein du système, on fait appel aux normes. Cela fait l'apparition des systèmes multi-agents normatifs vers lesquels notre travail est orienté.

Dans le chapitre suivant présentera un état de l'art sur le test des systèmes multi-agents (SMA).

Chapitre 2

Test des systèmes multi-agents

"Le savoir scientifique n'est pas absolu, mais socialement, culturellement, technologiquement et historiquement marqué, donc provisoire."

Steven Rose

Introduction

Afin de garantir la fiabilité et la satisfaction des utilisateurs, les développeurs doivent soumettre leur logiciel à des tests. Ces tests ne se limitent pas à la détection d'éventuelles anomalies, mais visent également à instaurer la confiance nécessaire pour une utilisation opérationnelle du produit logiciel et atteindre ainsi la qualité souhaitée. Pour les systèmes multi-agents (SMA), des techniques appropriées sont nécessaires pour évaluer les comportements autonomes des agents ainsi que leurs propriétés de distribution, sociales et délibératives spécifiques. Ce chapitre présente un état de l'art sur le test de logiciel en général et sur le test des systèmes multi-agents (SMA) en particulier.

1. Test de logiciel

1.1. Définition du logiciel

Le mot "logiciel" est la version française du terme anglais "Software". Il englobe tous les programmes et les procédures nécessaires pour faire fonctionner un système informatique. Parmi les différents types de logiciels, on peut citer les logiciels d'application qui sont conçus pour résoudre les problèmes spécifiques de l'utilisateur (tels que les progiciels, les tableurs, les traitements de texte, les graphiques, etc.), ainsi que les logiciels éducatifs ou didacticiels, les logiciels de jeux ou ludiciels, etc.

1.2. Test de logiciel

Il existe plusieurs définitions du test logiciel, qui permettent de comprendre pleinement cette notion. Voici quelques exemples [17-22] :

- Le test est un processus manuel ou automatique qui vise à vérifier si un système répond aux exigences de sa spécification ou à détecter des différences entre les résultats obtenus et ceux attendus.
- Tester, c'est exécuter un programme dans le but de trouver des anomalies ou des défauts.
- Le test consiste à évaluer un système ou un composant, de manière manuelle ou automatique, pour vérifier qu'il répond aux spécifications ou pour identifier les différences entre les résultats attendus et les résultats obtenus.
- Il s'agit d'une technique de contrôle qui permet de s'assurer que le comportement d'un programme est conforme à des données préétablies.

- Le test de logiciel est une activité de vérification dont le but est de repérer les erreurs ou les incohérences dans un programme. Il est important de préciser les références utilisées pour identifier ces inadéquations, telles que les spécifications du logiciel, les normes ou les règles concernant son code ou ses documents.
- Le test de logiciel est une étape du processus de développement qui suit les règles de l'assurance qualité. Il est mené une fois que la programmation est terminée et vise à minimiser les risques d'anomalies en détectant les erreurs potentielles et les défauts qui pourraient les causer, à l'aide de moyens manuels ou automatiques. Le test s'intéresse autant au code source qu'au comportement du logiciel.

1.3. Terminologie associée au test

Afin de comprendre la notion de test, il est important de connaître certaines notions telles que [23-26] :

- *Spécification* : En informatique, la spécification est un modèle qui décrit ce que le logiciel doit accomplir. Il existe trois types de spécification : informelle, semi-formelle et formelle.
- *Satisfaction* : Se réfère au fait qu'un programme respecte toutes les exigences de sa spécification.
- *Vérification* : Est le processus qui évalue un produit issu d'une activité de développement logiciel pour assurer sa correction et sa cohérence avec les normes et les produits fournis en entrée.
- *Validation* : Consiste à évaluer un logiciel afin de vérifier s'il répond aux exigences spécifiées.

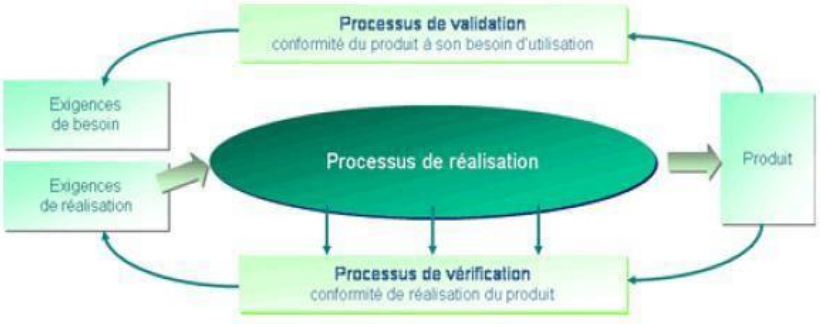

Figure 5. Concepts de vérification et validation [23].

- *Erreur* : Également appelée "error" en anglais, est une différence entre une valeur ou une condition spécifiée comme vraie ou théoriquement correcte, et la valeur ou la condition calculée, observée ou mesurée.
- *Défaut* : Ou "bug" ou "defect" en anglais, est une imperfection dans un composant ou un système qui peut empêcher celui-ci d'exécuter les fonctions requises, par exemple en raison d'une définition de données incorrecte. Un défaut peut entraîner la défaillance d'un composant ou d'un système.
- *Panne* : Ou défaillance, ou "failure" en anglais, correspond à l'incapacité d'un système ou d'un de ses composants à effectuer les fonctions demandées dans les conditions de performance spécifiées.

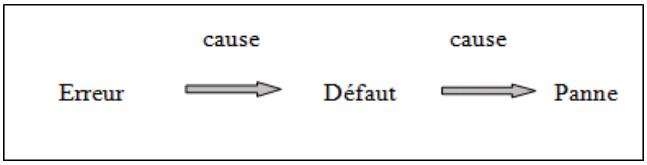

Figure 6. Relation entre défaillance / faute / erreur [26].

- *Cas de test* : Est un ensemble de valeurs d'entrée, de préconditions et de postconditions d'exécution qui sont développées pour atteindre un objectif particulier, comme exécuter un chemin spécifique dans un programme ou vérifier le respect d'une exigence spécifique.
- *Jeu de test* : Se compose des données d'entrée pour un cas de test, y compris les valeurs à saisir, les données réelles et la génération automatique. Il est important de noter que le même jeu d'essai peut être utilisé pour plusieurs cas de test.

1.4. Importance de test des logiciels

Le test est un élément crucial dans le développement d'un logiciel, qui représente une part importante des coûts (60 %). Il est essentiel pour garantir la qualité du produit final. Autrefois considéré comme une tâche secondaire, le test connaît aujourd'hui une révolution grâce à l'industrialisation de ses processus, la professionnalisation des métiers du test, l'arrivée à maturité d'une chaîne outillée allant des exigences au référentiel de tests, ainsi que la mise en place de centres de service dédiés aux activités de test. L'objectif du test est de vérifier le logiciel en utilisant des données similaires à celles du monde réel, afin de détecter les anomalies et les erreurs éventuelles [27].

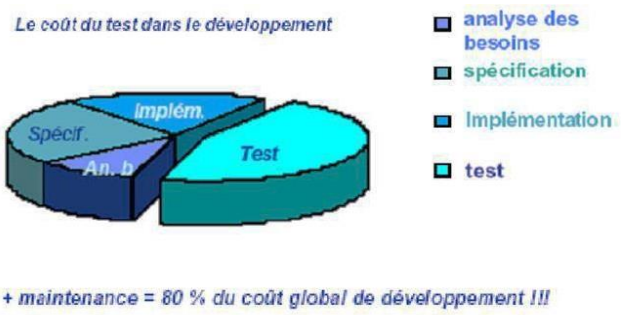

Figure 7. Coût du test dans le développement [26].

1.5. Objectif du test

Le but du test n'est pas de déterminer la source des erreurs, de les corriger ou de prouver que le programme est sans faute. Son objectif est de révéler les erreurs dans le logiciel, de confirmer leur existence, mais pas leur absence. Le test a pour but de détecter les défauts avant qu'ils ne provoquent une défaillance du système et de garantir que le logiciel testé atteigne un niveau de qualité acceptable après la correction des défauts identifiés [26].

1.6. Difficulté du test

Le test exhaustif est une méthode de vérification de logiciels qui consiste à tester toutes les exécutions possibles d'une application. Cependant, cette méthode est souvent impossible à réaliser, notamment en test fonctionnel où l'ensemble des données est infini ou de très grande taille. De même, en test structurel, le parcours du graphe de flot de contrôle conduit à une forte explosion combinatoire. Par conséquent, le test ne peut être qu'une méthode de vérification partielle de logiciels, dont la qualité dépend de la pertinence des données de test choisies. En outre, le test est un processus destructif qui vise à trouver des erreurs, contrairement à l'activité de programmation qui est un processus constructif visant à établir des résultats corrects. Les erreurs peuvent être dues à des incompréhensions des spécifications ou de mauvais choix

d'implantation. Ainsi, l'activité de test s'inscrit dans le contrôle de qualité et est indépendante du développement. L'exécution des tests peut être automatisée on peut écrire des programmes qui permettent d'enchaîner l'exécution de plusieurs tests, et qui vérifient que le résultat est correct [28].

Dans l'industrie du test, l'automatisation est souvent limitée à l'exécution des tests et à la comparaison des résultats avec les résultats attendus, avec parfois une mesure de la couverture. Les utilisateurs doivent élaborer les cas de test et les données de test, éventuellement à l'aide d'outils de préparation de scripts de test. Cela peut limiter le nombre de tests effectués en raison du coût et de la disponibilité des ingénieurs compétents, et la couverture des tests peut être difficile à évaluer. Cependant, la génération automatique de données de test et d'un oracle permet une sélection de cas de test basée sur des critères justifiés par des hypothèses explicites [23].

2. Test des systèmes multi-agents

Le test des SMA est une tâche complexe et délicate, en raison des caractéristiques uniques des agents de logiciel qui rendent difficile l'application des méthodes de test existantes. Les questions liées à la communication et à la coordination ajoutent à la difficulté, car ces dispositifs sont difficiles à modéliser, programmer et tester. Le reste de cette section examine en détail les enjeux liés au test des systèmes multi-agents.

2.1. Difficultés de test des SMA

Le test des SMA devient de plus en plus difficile en raison de plusieurs raisons [29] :

- La complexité des applications augmente en raison de leur nature distribuée, avec plusieurs agents fonctionnant de manière autonome et concurrente.
- Les agents manipulent une grande quantité de données, chacun ayant ses propres objectifs et buts.
- L'effet non reproductible se réfère au fait qu'il n'est pas certain que deux exécutions du système, même avec les mêmes données en entrée, aboutissent à un état identique. En conséquence directe, il peut être très difficile, voire impossible, de localiser une erreur, car il est peu probable de reproduire l'exécution qui a révélé l'erreur [30].
- Ils sont non déterministes, car il est impossible de prédire toutes les interactions d'un agent pendant son exécution.
- Les agents utilisent principalement la transmission de messages plutôt que l'appel de méthodes, ce qui rend les approches de test orientées objet inapplicables directement.
- Les agents sont autonomes et coopèrent avec d'autres agents, ce qui leur permet de fonctionner correctement de manière individuelle, mais de manière imprécise au sein d'une communauté.
- Un agent mobile est une catégorie spécifique d'agent qui peut se déplacer d'un endroit à un autre tout en continuant son exécution. Cette capacité de mobilité pose de nouveaux défis.

Par conséquent, pour tester les SMA, il est nécessaire de développer de nouvelles méthodes de test qui prennent en compte leur nature spécifique. Ces méthodes doivent être efficaces et appropriées pour évaluer les comportements autonomes des agents.

2.2. Niveaux de test des SMA

Les techniques de test traditionnel ne peuvent pas être appliquées directement à un système multi-agents. De plus, les spécifications de ces systèmes autonomes et concurrents sont souvent incomplètes, incohérentes voire incorrectes. Pour tester un tel système, il est nécessaire de le considérer commecomposé de différents niveaux d'abstraction.

Le test des SMA se compose de cinq niveaux, comme proposé dans [31] et [32] : unité, agent, intégration, système, et acceptation. Les objectifs de test et les activités de chaque niveau sont décrits comme suit :

2.2.1. Test unitaire

Il est important de tester tous les éléments constitutifs d'un agent, y compris les blocs de code, lesdifférentes unités telles que les buts, les plans, la base de connaissances, le moteur de raisonnement, la spécification des règles, etc. Il est crucial de vérifier que chacune de ces unités fonctionne comme prévu.

2.2.2. Test d'agent

Il convient de tester l'intégration des différents modules au sein d'un agent, ainsi que ses capacités à atteindre ses objectifs et à percevoir et agir sur l'environnement.

2.2.3. Test d'intégration ou de groupe

Il convient de tester plusieurs aspects dans le cadre de l'interaction des agents, tels que le protocole de communication, la sémantique, ainsi que leur interaction avec l'environnement et l'intégration avec les ressources partagées. Il est également important d'appliquer des règlements et d'observer les propriétés émergentes et les comportements collectifs ; s'assurer qu'un groupe d'agents et les ressources environnementales travaillent de manière harmonieuse et efficace.

2.2.4. Test de système ou de société

Teste le SMA en tant que système opérant dans un environnement, en évaluant les propriétés émergentes et macroscopiques prédites du système dans son ensemble. Évalue également les propriétés de qualité que le système doit atteindre, telles que l'adaptabilité, la fiabilité, la tolérance aux pannes et les performances.

2.2.5. Test d'acceptation

Il convient de tester le SMA dans l'environnement d'exécution du client afin de vérifier qu'il atteint les objectifs du dépositaire, avec la participation de ces derniers.

2.3. Aperçu sur le test des systèmes multi-agents

Il existe peu d'écrits qui décrivent le processus de test des systèmes multi-agents. Cette section se penche sur les travaux récents et en cours sur le test de ces systèmes, en les classant selon les catégories précédentes pour faciliter la compréhension des recherches dans ce domaine. Nous commençons par résumer les contributions des travaux qui se concentrent principalement sur un niveau de test particulier, avant de présenter les travaux qui couvrent plusieurs niveaux de test.

2.3.1. Niveau unitaire

- ***Zhang et al. [34, 35, 36]***

Ce travail présente un cadre pour automatiser les tests unitaires des systèmes d'agent. La méthode de test utilisée est basée sur des modèles qui sont des objets de conception issus de la méthodologie de conception d'agent. La méthodologie choisie est PROMETHEUS, qui peut être étendue à d'autres méthodologies

utilisant des concepts similaires [37]. L'outil utilisé pour intégrer les méthodes de test est l'outil de conception de PROMETHEUS (PDT).

- ***Ekinci et al. [38]***

Ce travail se concentre sur les buts d'agent qui sont considérés comme les plus petites unités testables dans les SMA. Il propose une méthode de test des buts qui consiste à les décomposer en trois sous-buts : installation, but sous test et affirmation. Les deux premiers sous-buts préparent les conditions préalables et vérifient les postconditions, tandis que le troisième sous-but teste le but sous test.

2.3.2. Niveau d'agent

- ***Lam et Barber [39]***

Ce travail présente un processus semi-automatisé visant à comprendre les comportements des agents logiciels. Cette approche s'inspire de la façon dont un utilisateur humain, tel qu'un testeur, aborde la compréhension d'un logiciel : elle consiste à construire et affiner une base de connaissances sur les comportements des agents, puis à l'utiliser pour vérifier et expliquer leurs actions lors de l'exécution.

- ***Nunez et al. [40]***

Ce texte présente un cadre formel pour spécifier le comportement des agents autonomes de commerce électronique. Les comportements souhaités des agents testés sont décrits à l'aide d'un nouveau formalisme appelé la machine d'état de service, qui représente les préférences des utilisateurs dans ses états. Deux méthodologies de test sont proposées pour vérifier si l'exécution d'un agent spécifique est conforme aux attentes (c'est-à-dire un test de conformité) : une active et une passive. Dans l'approche de test active, un test est utilisé pour chaque agent sous test (un agent spécial) qui prend les spécifications formelles de l'agent pour faciliter l'atteinte d'un état spécifique. La trace opérationnelle de l'agent est ensuite comparée aux spécifications pour détecter d'éventuels défauts. D'un autre côté, les auteurs suggèrent également l'utilisation du test passif, qui consiste à observer les agents sous test sans les simuler, contrairement au test actif. Les traces invalides, si elles existent, peuvent ainsi être identifiées en se référant aux spécifications formelles des agents.

- ***Coelho et al. [41]***

Ce document s'inspire de JUnit [42] et vise à fournir un cadre pour les tests d'unité des SMA en utilisant des agents Mock. L'objectif principal est de tester les rôles des agents. Pour ce faire, des agents Mock ont été manuellement implémentés afin de simuler la communication avec l'agent à tester. Chaque agent Mock correspond à un rôle d'agent spécifique.

- ***Nguyen et al. [43]***

Ce travail utilise les ontologies d'interaction d'agent pour définir la signification des interactions entre les agents. Ces ontologies permettent de (i) générer des entrées de test, (ii) guider l'exploration de l'espace d'entrée pendant la génération, et (iii) vérifier que les messages échangés entre les agents respectent l'ontologie d'interaction définie. Un ensemble de règles de génération a été établi et, en utilisant un cadre de test automatisé, il est possible de tester un agent particulier de manière intensive avec un grand nombre de cas de test variés.

- ***Nguyen et al [44]***

La présence d'agents autonomes rend les tests plus complexes. En effet, ces derniers peuvent réagir de différentes manières à des entrées identiques en raison de leurs objectifs et de leurs connaissances variables. Le test d'agents autonomes nécessite donc une procédure couvrant un large éventail de contextes de cas de test, incluant les plus difficiles. Nguyen et ses collègues ont développé et évalué une approche pour tester les agents autonomes en utilisant l'optimisation évolutionnaire pour créer des scénarios de test difficiles. Dans leur étude, les auteurs ont proposé une approche systématique pour évaluer la qualité des agents autonomes. Tout d'abord, les exigences du système sont représentées sous forme de mesures de qualité, et les seuils correspondants sont utilisés comme critères de test. Les agents autonomes doivent répondre à ces critères pour être considérés comme fiables. Des fonctions physiques sont ensuite définies pour représenter les objectifs de test, et guident la méthode de génération de tests évolutionnaires pour produire des scénarios de test automatiquement.

2.3.3. Niveau d'intégration

- ***Serrano et Botia [45], Serrano et al. [46]***

Lorsqu'on teste les SMA, l'une des difficultés réside dans leur évolutivité, due au grand nombre d'agents et d'interactions possibles. Cela rend l'application de méthodes de test classiques très complexe. Pour résoudre ce problème, ACLAnalyser utilise des techniques de Data mining pour analyser les exécutions de SMA (initialement sur la plate-forme d'agents JADE) et découvrir des modèles d'apparition au niveau social du système. Par exemple, les communautés d'agents avec des liens d'interaction forts peuvent être détectées grâce aux fonctionnalités de regroupement d'ACLAnalyser.

2.3.4. Plusieurs niveaux de test

- ***Tiryaki et al. [47]***

Ce travail présente une méthode de développement de SMA basée sur les tests de conduite, qui favorise une construction itérative et incrémentale des SMA. Pour soutenir cette approche, un cadre de test appelé SUnit a été développé en utilisant JUnit [42] et Seagent [48]. Ce cadre permet l'écriture de tests pour les comportements des agents ainsi que pour les interactions entre les agents.

- ***Gomez-Sanz et al. [49]***

Ce travail se focalise sur l'agent et le niveau d'interaction de test. Le kit de développement d'INGENIAS offre des outils pour tester des SMA. Cette méthode repose sur l'approche modèle-conduite d'INGENIAS, qui permet de spécifier des suites de test en modélisant le SMA. On peut ainsi spécifier des déploiements de test, des tests d'interaction, ainsi que l'inspection de l'état mental des agents. Les modèles de SMA sont convertis en code dans le cadre d'agent d'INGENIAS, qui est ensuite exécuté sur la plate-forme d'agent de JADE. Cela permet aux développeurs d'exécuter des tests et de suivre la progression des opérations d'organisation, des interactions et des états mentaux des agents. Les suites de test peuvent être automatisées, ce qui facilite le développement agile basé sur les modèles en permettant des itérations rapides avec des tests de validation.

- ***Nguyen et al. [50]***

Le document décrit une méthodologie de test appelée Goal-Oriented Software Testing (GOST), qui utilise une approche basée sur l'analyse et la conception orientées vers les buts pour créer des suites de test à tous les niveaux, de l'unité à l'acceptation. Cette méthode fournit un modèle de processus de test qui relie les objectifs aux cas de test explicites et permet de dériver systématiquement des cas de test à partir de l'analyse des objectifs. Cela peut aider à détecter rapidement des problèmes, évitant ainsi la mise en place de spécifications incorrectes. De plus, le GOST est accompagné d'un outil qui facilite la génération de cas de test, la spécification et l'exécution.

Parmi les approches de test des systèmes multi-agents citées dans cette section 2 de ce chapitre, il n'y a aucune qui assure la fiabilité de ce genre de systèmes. Cela nous a conduit dans un premier temps à poser la question : **Comment peut-on assurer la fiabilité des SMA, plus particulièrement, les SMA normatifs ?** Avant de répondre à cette question, nous sommes obligés de découvrir des notions telles que la fiabilité, le test de fiabilité, … dans la littérature. Ses notions vont être présentées dans la section suivante de ce chapitre. Second, et pour répondre à notre question, nous allons s'inspirer de la littérature pour proposer une nouvelle approche de test de fiabilité des SMA normatifs. Cette approche sera proposée dans le prochain chapitre.

3. Test de fiabilité

3.1. La fiabilité dans les logiciels

La fiabilité du logiciel est définie comme la probabilité de fonctionnement du logiciel sans défaillance pendant une période de temps déterminée dans un environnement particulier. Elle peut s'exprimer par [51]:

- La probabilité de non-défaillance pendant une durée t.
- Le taux de défaillance.
- Le temps moyen entre deux défaillances.
- La durée moyenne de fonctionnement après réparation.
- Le temps moyen de fonctionnement avant la première défaillance ou avant défaillance.

Si un produit logiciel fonctionne sans défaillance pendant une période donnée dans un environnement spécifié, il est alors appelé logiciel fiable.

3.2. Test de fiabilité des logiciels

Le test de fiabilité est une méthode de vérification de la performance d'un logiciel dans des conditions spécifiques et pour une durée prédéterminée. Cette technique permet de détecter les défauts de conception du logiciel qui doivent être corrigés afin d'assurer son fonctionnement optimal. Elle s'assure que le logiciel fonctionne correctement et sans interruption dans un environnement particulier, et garantit l'absence de bugs et d'erreurs dans la réalisation de ses objectifs [52].

Les objectifs des tests de fiabilité :

- Construire un produit robuste et fiable qui fonctionne parfaitement à chaque fois.
- Pour trouver les erreurs et les résoudre avant que le logiciel ne soit livré et diffusé.
- S'assurer que tous les clients sont satisfaits et que toutes ses exigences sont réalisées.
- Pour découvrir les problèmes de conception et de fonctionnalité du produit.
- Pour débloquer les derniers modèles de conception dans les tendances des défauts.

3.3. Modélisation de la fiabilité des logiciels

Parmi tous les modèles de fiabilité logicielle, Software Reliability Growth Models (SRGM) sont probablement les techniques les plus réussies dans la littérature, avec plus de 100 modèles existant sous une forme ou une autre, à travers des centaines de publications. Cependant, dans la pratique, les SRGM rencontrent des défis majeurs. Tout d'abord, les testeurs de logiciels suivent rarement le profil opérationnel pour tester le logiciel, alors ce qui est observé lors des tests logiciels peut ne pas être directement extensible pour une utilisation opérationnelle. Deuxièmement, lorsque le nombre d'échecs collectés dans un projet est limité, il est difficile de faire des prédictions de fiabilité statistiquement significatives. Troisièmement, certaines des hypothèses de SRGM ne sont pas réalistes, par exemple, les hypothèses selon lesquelles les défauts sont indépendants les uns des autres ; que chaque défaut a la même chance d'être détecté dans une classe ; et que la correction d'un défaut n'introduit jamais de nouveaux défauts [53]. Néanmoins, les revers ci-dessus peuvent être surmontés avec des moyens adaptés. Compte tenu d'un processus approprié de collecte de données pour éviter une invalidation drastique des hypothèses du modèle, il est généralement possible d'obtenir des estimations de fiabilité et de savoir que ces estimations sont exactes.

Bien que certains SRGM historiques aient été largement adoptés pour prédire la fiabilité des logiciels, les chercheurs croient qu'ils peuvent encore améliorer la prédiction de l'exactitude de ces modèles en ajoutant d'autres éléments importants, facteurs qui affectent la qualité finale du logiciel [54,55,56]. Entre autres, la couverture de code est une métrique couramment engagée par les testeurs de logiciels, comme cela indique à quel point un ensemble de tests exécute un système logiciel sous test, influençant ainsi le résultat mesure de fiabilité. En incorporant l'effet de la couverture de code sur la fiabilité dans les modèles traditionnels de fiabilité de logiciels, [54] propose une technique utilisant la mesure de la couverture temporelle et du code pour la prédiction de fiabilité. Cette technique réduit le temps d'exécution d'un facteur paramétré lorsque le cas de test n'augmente ni la couverture du code ni ne provoque une panne. Ces modèles, dits non homogènes ajustés, Les modèles de processus de Poisson (NHPP), ont été montrés empiriquement pour obtenir des prédictions plus précises que ceux d'origine.

Dans la littérature, plusieurs modèles ont été proposés pour déterminer la relation entre le nombre de pannes/défauts et la couverture de test obtenue, avec diverses diffusions. [56] suggère que cette relation est une variante de la distribution de Rayleigh, tandis que [55] montre qu'il peut être exprimé comme une formule exponentielle logarithmique, basée sur l'hypothèse que les deux fautes, la couverture et la couverture de test, suivent le modèle de croissance NHPP par rapport au temps d'exécution. Plus de métriques peuvent être incorporées pour explorer plus avant cette nouvelle avenue de modélisation.
Bien qu'il existe un certain nombre de modèles de fiabilité réussis, ce sont généralement des modèles basés sur des mesures qui sont employés isolément au stade ultérieur du processus de développement logiciel. Les premiers modèles logiciels de prédiction de fiabilité sont souvent trop peu substantiels, rarement exécutables, insuffisamment formels pour être analysables et généralement non liés au système cible. Leur impact sur la fiabilité résultante est donc modeste. Il existe actuellement un besoin pour un modèle fiable de fiabilité logicielle de bout en bout qui peut être directement lié à la prédiction de la fiabilité dès le début, afin d'établir une procédure systématique pouvant être certifiée, généralisée et affinée.

3.4. Méthodes et outils

En plus de la modélisation de la croissance de la fiabilité des Logiciels, de nombreuses autres méthodes sont disponibles pour l'ingénierie de fiabilité des logiciels (Software Reliability Engineering). Nous donnons quelques exemples de ces méthodes et outils :

Les arbres de défaillance fournissent un cadre graphique et logique pour une analyse systématique des modes de défaillance du système. Les ingénieurs en fiabilité logicielle peuvent les utiliser pour évaluer l'impact global des défaillances logicielles sur un système, ou pour prouver que certains modes de défaillance ne se produiront pas. S'ils peuvent se produire, la probabilité d'occurrence peut également être évaluée. Les modèles d'arbres de défaillances fournissent donc un cadre de modélisation informatif pouvant être engage pour comparer différentes alternatives de conception ou systèmes architectures en matière de fiabilité.

En particulier, ils ont été appliqués à la fois à la tolérance aux pannes et à la panne systèmes intolérants (c'est-à-dire non redondants). Puisque cette technique provient de systèmes matériels et a été étendue aux systèmes logiciels, elle peut être employée pour fournir un schéma de modélisation unifié pour la co-conception matériel/logiciel. La modélisation de la fiabilité pour les interactions matériel-logiciel est actuellement un domaine de recherche intensive [57].

De plus, des techniques de simulation peuvent être fournies à des fins SRE. Elles peuvent produire des observables d'intérêt pour l'ingénierie de la fiabilité, y compris des quantités discrètes à Valeur entière qui se produisent au fil du temps.

Une approche de simulation produit des artefacts dans un Environnement logiciel réel en fonction de facteurs et influences censées caractériser ces entités au sein d'un contexte donné [58]. Les artefacts et l'environnement sont autorisés à interagir naturellement, après quoi le flux d'occurrences d'activités et d'événements est observé. Cette simulation basée sur des artefacts permet de définir des expériences visant à examiner la nature des relations entre les pannes logicielles et d'autres métriques logicielles, telles que la structure du programme, les caractéristiques des erreurs de programmation et les stratégies de test.

Il est suggéré que la mesure dans laquelle la fiabilité dépend uniquement de ces facteurs peut être mesurée en générant des programmes aléatoires ayant les caractéristiques données, puis en observant leur statistique d'échec.

Une autre approche de simulation de fiabilité [59] produit des imitations chronologiques d'activités liées à la fiabilité et des événements. Les mesures de fiabilité d'intérêt pour le logiciel et le processus sont modélisées paramétriquement dans le temps. La clé de cette approche est une architecture basée sur les taux dans laquelle les phénomènes se produisent naturellement au fil du temps, comme contrôlés par leurs fréquences d'occurrence. Ces fréquences dépendent des métriques logicielles de conduite telles que le nombre de défauts jusqu'ici exposés ou encore subsistants, la défaillance criticité, le niveau de main-d'œuvre, l'intensité des tests et le temps d'exécution du logiciel. La simulation d'événement basée sur le taux est un exemple d'une forme de modélisation appelée dynamique des systèmes, dont la particularité est que les observables sont des événements discrets se produisant au hasard dans le temps. Puisque de nombreux modèles de croissance de la fiabilité des logiciels sont également basés sur le taux (en termes de risque logiciel), les processus sous-jacents supposés par ces modèles sont fondamentalement identiques à

la simulation de fiabilité basée sur le débit. En général, les simulations permettent d'enquêter sur des questions trop difficiles à répondre analytiquement, et sont donc plus souples et plus puissantes.
Divers outils de mesure SRE ont été développés pour la collecte de données, l'analyse de fiabilité, l'estimation des paramètres, l'application du modèle et la fiabilité simulation. Toute amélioration majeure du SRE est probablement concentrée sur de tels outils. Nous devons fournir des outils et des environnements qui peuvent aider les développeurs de Logiciels à construire un logiciel fiable pour différentes applications. La partition d'outils, d'environnements et de techniques qui sera engagé devrait refléter l'emploi approprié des meilleures pratiques SRE actuelles.

3.5. Test de fiabilité et profils opérationnels

Le profil opérationnel est une caractérisation quantitative de la façon dont un système sera utilisé sur le terrain par les clients. Il aide à planifier les activités de test, à générer des cas de test et à sélectionner des exécutions de test. En allouant des ressources de développement et de test aux fonctions sur la base de leur utilisation, la fiabilité de l'ingénierie des Logiciels peut ainsi être planifiée avec productivité et considérations économiques à l'esprit. L'utilisation d'un profil opérationnel pour guider les tests du système garantit que si les tests sont terminés et que le logiciel est expédié en raison de contraintes de calendrier impératives, les opérations les plus utilisées auront reçu le plus de tests, et le niveau de fiabilité sera le maximum qui est pratiquement réalisable pour le temps de test donné.
Aussi, en guidant les tests de régression, le profil a tendance à trouver, parmi les défauts introduits par les changements, ceux qui ont le plus d'effet sur la fiabilité. Exemples des avantages de l'application de profils opérationnels peuvent être trouvés dans un certain nombre de projets industriels [60].
Bien qu'une amélioration significative puisse être obtenue en employant des profils opérationnels en régression ou en test du système, des défis existent toujours pour cette technique. Tout d'abord, certains profils opérationnels d'applications sont difficiles à développer, surtout pour certains systèmes Logiciels distribués, par exemple les services Web.
De plus, à la différence de ceux du matériel, le fonctionnement des profils de Logiciels ne peut pas être dupliqué afin d'accélérer les tests, car le comportement de défaillance du logiciel dépend grandement de sa séquence d'entrée et de son statut interne. Pendant les tests unitaires, différentes unités de logiciel peuvent être testées en même temps. Cette approche n'est donc pas applicable dans les tests de système ou la régression d'essai. En conséquence, apprendre à gérer les profils opérationnels et les dépendances au sein des profils opérationnels sont les deux problèmes majeurs techniques des profils opérationnels.

3.5.1. Travaux de la génération du profil opérationnel

Plusieurs travaux liés au développement de SOP peuvent être trouvés dans la littérature. Musa a introduit SOP dans [60] et [61]. Le processus de développement contient cinq étapes, à savoir le profil client, le profil utilisateur, le profil de mode système, le profil fonctionnel et le profil opérationnel. Il a illustré son approche par une étude de cas de système de commutation de télécommunication. Une méthode conçue pour créer des profils opérationnels en décomposant un OP en deux composants : un profil de configuration et un profil d'utilisation est présentée dans [62]. Ce document fournit des exemples du projet dans lequel cette méthode a été utilisée pour la première fois (AT&T Private Branch Exchange). Cette approche est particulièrement utile lorsque les clients, l'utilisation et les configurations du produit sont variés et complexes.

Dans [63], une méthode structurée pour développer un profil opérationnel pour les utilisateurs interactifs est expliquée et illustrée par un exemple. Elle montre également l'utilisation d'un outil de sélection de cas de test qui facilite la tâche du testeur en utilisant comme entrée les informations obtenues lors de l'élaboration du profil opérationnel. Un autre travail propose une méthodologie de raffinement pour la génération de profils opérationnels plus précis qui représentent véritablement les divers modèles d'utilisation des clients [64]. L'analyse de regroupement prend en charge la méthodologie de raffinement pour identifier des groupes de clients ayant des caractéristiques similaires. Le modèle de profil opérationnel est étendu dans [65]. La version étendue est composée d'un profil de processus, d'un profil structurel et d'un profil de données en plus du profil opérationnel. Le profil de processus décrit les processus et les fréquences avec lesquelles ils se produisent dans une application utilisateur typique. Le profil de structure est la structure du système sur lequel l'application est en cours d'exécution. Les valeurs des entrées dans l'application d'un ou plusieurs utilisateurs présentent un profil de données.

Un développement du profil opérationnel systématique pour les composants logiciels est présenté dans [66]. Le procédé utilise à la fois des hypothèses d'utilisation prévues et des données d'utilisation pour découvrir une structure d'utilisation, une distribution d'utilisation et des caractéristiques de paramètres. Dans [67], un logiciel qui avait subi le cycle de test programmé normal de l'entreprise auparavant est testé à travers un profil opérationnel. Un SOP est développé et les cas de test sont attribués.

Un cadre multi-agents pour générer automatiquement un profil opérationnel pour les systèmes distribués est présenté dans [68]. Ce travail propose également de nouvelles métriques composées pour déterminer la criticité des opérations. Un cadre pour développer une allocation de cas de test basée sur un SOP en utilisant la logique floue est proposé dans [69]. Deux études de cas sont présentées pour montrer l'applicabilité du cadre proposé. Une autre méthode basée sur la conception uniforme est proposée deux ans plus tard dans [70] pour développer SOP. Dans [71], un examen d'un SOP, son extension, sa mise en œuvre et ses lacunes sont présentés, tandis que [72] est consacré à l'enquête, l'analyse et la classification des profils opérationnels qui caractérisent le type et la fréquence des entrées logicielles. Dans [73], une méthodologie de développement de SOP de manière efficace est proposée avec quelques modifications dans l'approche de Musa. Pour rendre efficaces le profil opérationnel, le profil d'utilisation et le profil de configuration, plus de profils ont été ajoutés dans le modèle existant. Le profil d'utilisation montre que les utilisateurs ne sont pas tous pareils dans leur proportion d'utilisation, tandis que le profil de configuration définit la façon dont le client configure son système.

Une autre recherche doctorale est présentée dans [74] qui étudie l'utilisation des SOP comme ressource pour assurer et améliorer la qualité du logiciel du point de vue des utilisateurs. La recherche propose une stratégie dans laquelle les utilisateurs utilisent le SOP pour évaluer et adapter une suite de tests existante et pour classer l'impact des défauts dans le fonctionnement du logiciel, contribuant à la tarification des défauts. Une approche de développement axée sur le comportement [75] est utilisée comme source d'information pour la génération semi-automatisée du profil opérationnel. L'approche proposée a été évaluée sur le logiciel de réseau social Diaspora, un projet open-source.

Récemment, une nouvelle méthodologie [76] a été publiée pour développer un profil opérationnel pour les systèmes multi-agents normatifs (NorMAS-OP). Cette approche implique jusqu'à cinq étapes, les étapes sont les suivantes : trouver le profil de rôle client NorMAS, identifier le profil de rôle utilisateur NorMAS, définir le profil de mode NorMAS, déterminer le profil d'objectif NorMAS et déterminer le profil opérationnel NorMAS.

Discussion

Bien que la recherche sur les systèmes multi-agents normatifs soit intéressante, le développement d'applications basées sur ces systèmes demeure peu solide. La tâche importante d'assurer la qualité de ces systèmes à travers l'activité de test n'est pas bien couverte. En effet, la littérature propose très peu de méthodes pour tester ces systèmes, laissant plusieurs questions sans réponse. Par exemple, **comment peut-on s'assurer de la fiabilité des systèmes multi-agents pour qu'ils répondent à leurs spécifications fonctionnelles ?** Cette question reste un objectif de recherche.

Les approches existantes pour tester les systèmes multi-agents ont certainement apporté des réponses significatives à diverses problématiques liées aux SMA. Cependant, sauf le travail proposé dans [76], ces approches ne prennent pas en compte les spécificités des SMA normatifs.

Conclusion

La fiabilité du logiciel fait référence à la capacité d'un système à exécuter sa fonction prévue dans des conditions spécifiées pendant une période de temps spécifiée. La première étape critique du processus de test de fiabilité du logiciel consiste à créer un profil opérationnel du logiciel (SOP). Plusieurs méthodologies de création de SOP ont été proposées. Néanmoins, presque toutes les études proposées ont négligé le caractère unique des nouveaux paradigmes logiciels, malgré le fait que ceux-ci se distinguent généralement par leurs propres concepts et méthodologies. L'un de ces paradigmes logiciels est celui des systèmes multi-agents. Récemment, un travail utile [76] a introduit le test de fiabilité par une méthodologie spécifique pour la création du profil opérationnel pour les SMA normatifs. Dans le cadre de notre travail, et en se basant sur [76], nous allons proposer une nouvelle approche de test de fiabilité des SMA normatifs.

Chapitre 3
L'approche proposée

" La rapidité des esprits scientifiques rattrapera toujours l'imagination des artistes."

Fabien Blanchot

Introduction

La fiabilité du logiciel est un attribut important de la qualité du logiciel. Comme les logiciels deviennent de plus en plus complexes, la fiabilité des logiciels est devenue difficile à atteindre. Dans la littérature, de nombreuses études ont été proposées pour les tests de fiabilité des logiciels. Cependant, dans le cadre des systèmes multi-agents, cette activité est complètement omise. Étant donné que les systèmes logiciels multi-agents sont largement utilisés pour mettre en œuvre des systèmes logiciels complexes, tester leur fiabilité est une tâche inévitable.

Dans la première partie de ce chapitre, nous allons proposer une nouvelle approche de test de fiabilité pour les systèmes multi-agents. Elle est basée sur les deux mesures bien connues : MTTF (Mean Time To Failure) et MTBF (Mean Time Between Failures) tout en prenant en compte les quatre métriques de qualité (autonomie, sociabilité, modularité et adaptabilité) liées à la fiabilité du logiciel définies dans le Modèle de qualité ISO 25010. L'approche proposée présente un compromis entre les solutions traditionnelles basées sur des modèles de fiabilité et l'assurance qualité des logiciels multi-agents. La deuxième partie du chapitre est consacré pour l'automatisation de la méthodologie de génération du profil opérationnel pour les SMA normatifs [76], qui est une étape cruciale dans notre approche. La génération automatique du profil opérationnel est illustrée par l'étude de cas mentionner dans [76].

1. Une nouvelle approche pour tester la fiabilité des SMA normatifs

L'ingénierie de la fiabilité des logiciels (Software reliability engineering SRE) se concentre sur les techniques d'ingénierie pour développer et maintenir des systèmes logiciels dont la fiabilité peut être évaluée quantitativement. En respectant le processus SRE présenté dans [77], nous introduirons un processus complet adapté pour les tests de fiabilité des SMA. Comme il est montré dans la figure 8, nous divisons le processus en sept étapes : A savoir, déterminer les objectifs de fiabilité du MAS, développer le profil opérationnel du MAS, effectuer les tests du MAS, collecter/enregistrer les données de défaillance, mesurer la fiabilité du MAS, calculer les mesures de qualité du MAS et valider le niveau de fiabilité du MAS.

1.1. Déterminer les objectifs de fiabilité du MAS

Les objectifs de fiabilité doivent être clairement définis de manière quantitative. Si les objectifs de fiabilité ont été spécifiés par le client, ils doivent être utilisés. Sinon, un ensemble de mesures de fiabilité appropriées est nécessaire pour exprimer quantitativement la fiabilité du logiciel [78]. Pour déterminer les objectifs de fiabilité du MAS, nous utiliserons des mesures traditionnelles telles que MTTF (Mean Time To Failure) et

MTBF (Mean Time Between Failures) combinées à quatre métriques de qualité (autonomie, sociabilité, modularité et adaptabilité) affectant la fiabilité des logiciels définis dans la norme ISO 25010 modèle de qualité. La sixième étape du processus de test de fiabilité présente plus de détails sur les mesures de qualité.

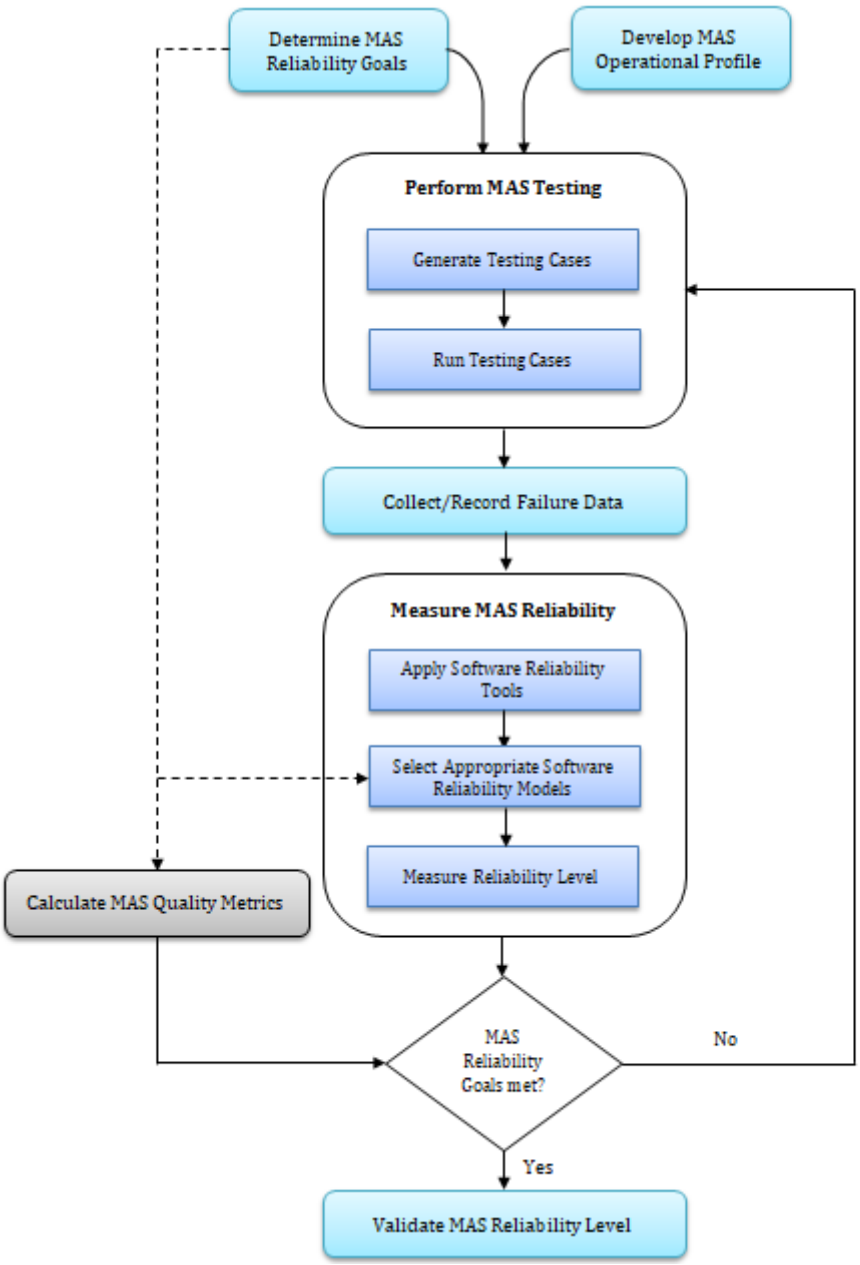

Figure 8. Approche proposée pour le test de fiabilité des SMA normatifs.

1.2. Développer le profil opérationnel du MAS

Le profil opérationnel (OP) [60] [61] est essentiel dans toute application SRE. Le profil opérationnel est l'ensemble des scénarios opérationnels du système et leurs probabilités associées. L'utilisation d'un profil opérationnel pour guider les tests garantit que si les tests sont terminés et que le logiciel est livré en raison de contraintes de calendrier impératives, les opérations les plus utilisées auront reçu le plus de tests et le niveau de fiabilité sera le maximum pratiquement réalisable pour le produit donné. Le profil opérationnel est développé pour décrire le fonctionnement du système logiciel et implique une ou plusieurs des cinq étapes suivantes : définir le profil du client, identifier le profil de l'utilisateur, définir le profil du mode

système, déterminer le profil fonctionnel et déterminer le profil opérationnel lui-même. Nous considérons les cinq étapes de [76] dans la construction du profil opérationnel des SMA.

1.3. Effectuer les tests du MAS

L'exécution d'un test à partir d'un profil opérationnel implique généralement les étapes suivantes : (1) développer le modèle de profil opérationnel et ses probabilités, (2) définir le plan de test avec les cas de test associés et (3) exécuter les cas de test. La norme IEEE 610 (1990) définit le cas de test comme "un ensemble d'entrées de test, de conditions d'exécution et de résultats attendus développés pour un objectif particulier, tel que l'exercice d'un chemin de programme particulier ou la vérification de la conformité à une exigence spécifique" [79]. Une fois le profil opérationnel MAS développé, les cas de test peuvent être générés avec un degré d'automatisation supérieur ou inférieur à partir du profil opérationnel. Ensuite, tous les cas de test générés précédemment seront traduits dans les scripts de test.

1.4. Collecter/Enregistrer les données de défaillance

Dans cette étape, les données de défaillance requises doivent être collectées et enregistrées correctement pendant le processus de test. Parmi les données de panne à collecter, il y a celles utilisées pour localiser et éliminer les pannes, telles que l'emplacement de la panne, le numéro de série du boîtier de test ; et ceux associés à la fiabilité, enregistrés pour cette dernière analyse de fiabilité, tels que le temps de défaillance et la gravité de la défaillance. Les défaillances détectées doivent être corrigées immédiatement dans les tests de fiabilité de croissance [80].

1.5. Mesurer la fiabilité du MAS

La mesure de la fiabilité est un élément essentiel du processus de test de fiabilité. Il détermine si un produit répond à ses objectifs de fiabilité et est prêt à être commercialisé [77]. Un ou plusieurs modèles de fiabilité sont appliqués pour estimer la fiabilité du système à partir des données de défaillance collectées au cours de l'étape de test du système. Au cours de cette étape du processus, nous devons sélectionner le modèle de fiabilité le plus approprié pour notre système. Etant donné le grand nombre de modèles, un problème de sélection se pose. Pour cela, plusieurs méthodes et classifications [81][82][83][84] ont été proposées afin de sélectionner un modèle approprié. En outre, il existe certains critères cités dans [78] par lesquels les modèles de fiabilité des logiciels peuvent être évalués. Généralement, il suffit de considérer les modèles pris en charge par les outils de fiabilité logicielle. Une fois que les modèles de fiabilité sont déterminés par les outils logiciels de fiabilité, nous devons utiliser ces modèles pour estimer ou prédire la fiabilité actuelle.

1.6. Calculer les mesures de qualité du MAS

Le logiciel fiable peut rester opérationnel même avec l'apparition d'éventuelles erreurs. Le principe de cette caractéristique est d'empêcher la propagation des erreurs afin d'en limiter les conséquences. Dans SMA, plusieurs techniques permettent de limiter la propagation des erreurs. Tout d'abord, si l'interaction est restreinte entre les agents, les agents défaillants potentiels ne peuvent pas affecter la fonctionnalité des autres agents. Par conséquent, l'interaction restreinte peut augmenter la fiabilité de l'ensemble du SMA. De plus, l'autonomie permet à l'agent de fonctionner sans l'intervention d'autres agents. De plus, un agent bien développé peut éviter les défaillances potentielles de l'une de ses parties. Un bon niveau de modularité limite la propagation des erreurs. Ainsi, un agent modulaire peut rester opérationnel si une erreur affecte

un module sans influence sur les autres modules. Dans d'autres cas, un agent peut utiliser l'adaptabilité pour changer son comportement, sa structure ou ses buts si une erreur empêche l'exécution du comportement initial ou l'atteinte des buts initiaux [85]. En ajoutant la sociabilité, ces quatre caractéristiques influencent la fiabilité du MAS et doivent être prises en compte lors du processus de test de fiabilité du MAS.

1.7. Valider le niveau de fiabilité du MAS

Les estimations de fiabilité résultant de la cinquième étape et les mesures de qualité évaluées à la sixième étape sont ensuite utilisées pour valider si les objectifs de fiabilité ont été atteints.

2. Génération automatique du profil opérationnel pour les SMA normatifs : étude de cas

Selon [60], l'élaboration d'un profil opérationnel comporte jusqu'à cinq étapes. Le même nombre d'étapes constitue la méthodologie proposée pour élaborer le profil opérationnel des systèmes multi-agents normatifs (NorMAS-OP). Comme le montre la figure 9, les étapes sont les suivantes : identifier le profil de rôle client du NorMAS, identifier le profil de rôle utilisateur du NorMAS, définir le profil de mode du NorMAS, déterminer le profil des objectifs du NorMAS et déterminer le profil opérationnel final du NorMAS.

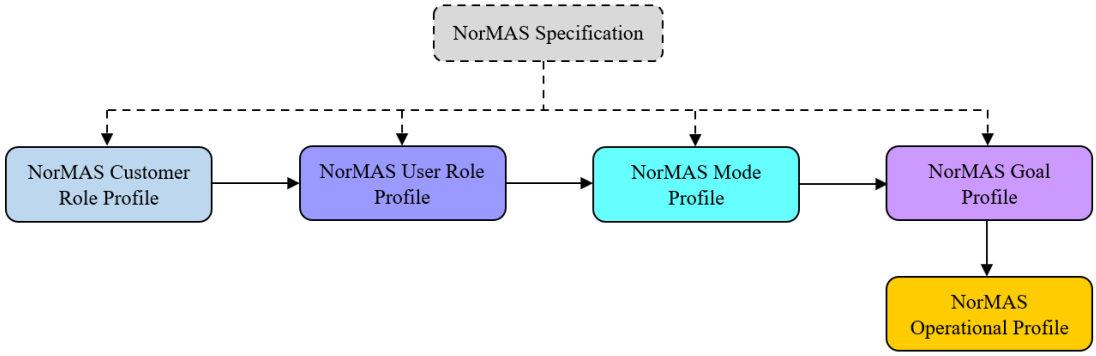

Figure 9. La méthodologie de développement du NorMAS-OP [76].

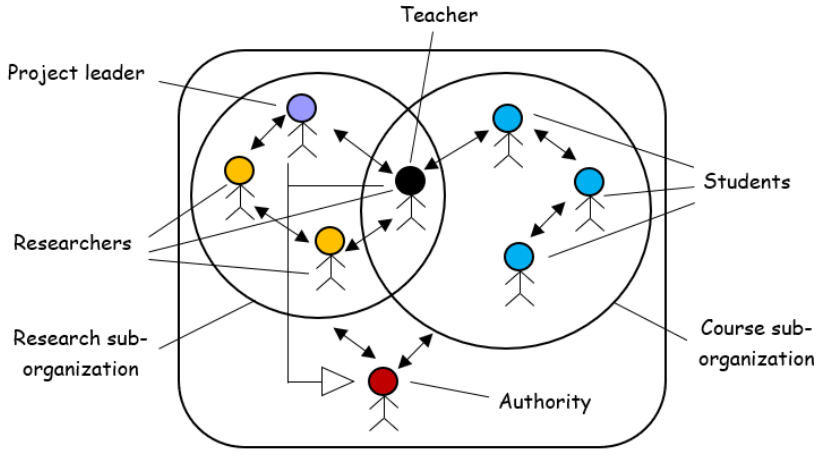

Figure 10. Le système « Université » [76].

Afin d'illustrer cette méthodologie, nous considérons le système « Université » comme un exemple de systèmes multi-agents normatifs présenté dans la figure 10 [76]. En fait, cette étude de cas est un exemple

typique de systèmes multi-agents normatifs. Il présente presque tous les concepts connexes de tels systèmes, comme les organisations, les rôles, les agents et les normes.

2.1. Spécification des normes du système universitaire

L'objectif global du système "Université" se décline en deux sous-objectifs : la recherche (Research) et l'enseignement (Course). Ces sous-objectifs sont divisés en un ensemble d'objectifs de base que les agents du système s'engagent à satisfaire. À leur tour, les objectifs de base sont regroupés en missions et distribués aux différents agents du système [76].

Tous les agents sont obligés, autorisés ou recommandés pour remplir leurs missions. Par exemple, l'Enseignant-Chercheur (Teacher-Researcher) est tenu de satisfaire à la mission (Mission-T) définie par les objectifs d'enseigner (teach), d'évaluer (evaluate), de rechercher (Research) et de respecter la réglementation (Respect regulations). Le même agent est autorisé pour occuper un poste supérieur (Occupy a senior position).

Le tableau 4 décrit les différentes normes auxquelles chaque rôle d'agent est attribué à une mission selon des normes de type (Obligation, Permission et Recommandation). L'administration, en tant qu'autorité, surveille l'application de ces normes. L'agent d'administration peut punir (Punish) un agent s'il ne satisfait pas à une norme d'obligation, et récompenser (Recompense) un autre s'il satisfait une norme de recommandation.

Tableau 2. Spécification normative du système universitaire [76].

Norm	Agent role	Type	Mission	Authority role
Nor1	Teacher-Researcher	Obligation	Mission-T = {goal1(Teach), goal2(Evaluate), goal3(Research), goal4(Respect regulations)}	Administration, Project leader
Nor2	Teacher-Researcher	Permission	Mission-T = {goal5(Occupy a senior position)}	Administration
Nor3	Student	Obligation	Mission-S = {goal6(Guided & Autonomous practice), goal7(Pass evaluation), goal4(Respect regulations)}	Teacher-Researcher, Administration
Nor4	Student	Recommendation	Mission-S = {goal8(Attend courses)}	Teacher-Researcher
Nor5	Researcher	Obligation	Mission-R = {goal9(Collaborate with researchers), goal10(Advance in research), goal11(Attend seminars), goal4(Respect regulations)}	Project leader
Nor6	Researcher	Permission	Mission-R = {goal1(Teach)}	Administration
Nor7	Project leader	Obligation	Mission-Pl = {goal3(Research), goal12(Make industrial contacts), goal13(Sign contracts), goal14(Propose approach), goal15(Supervising researcher), goal4(Respect regulations)}	Administration
Nor8	Project leader	Permission	Mission-Pl = {goal5(Occupy a senior position)}	Administration
Nor9	Administration	Permission	Mission-A = {goal16(Punish), goal17(Recompense)}	-

2.2. Étape 1 : Profil de rôle client (Customer role profile)

Trouver le profil du rôle du client est la première étape du processus de développement. Un client est un acteur, groupe d'acteurs qui vont acquérir le système. Un groupe de rôles clients est défini par un ensemble de rôles clients qui utiliseront le système de la même manière. L'ensemble des groupes de rôles clients et leur probabilité d'occurrence associée définissent le profil client. Dans le cas d'un système universitaire, le profil de rôle client n'est pas utilisé car il n'y a pas de type de rôle client. Par conséquent, la probabilité d'occurrence qui devrait être associée au profil de rôle client est de 1.

La figure 11 montre l'interface d'accueil de notre outil de génération automatique du profil opérationnel, alors que la figure 12 présente l'interface de profil de rôle client.

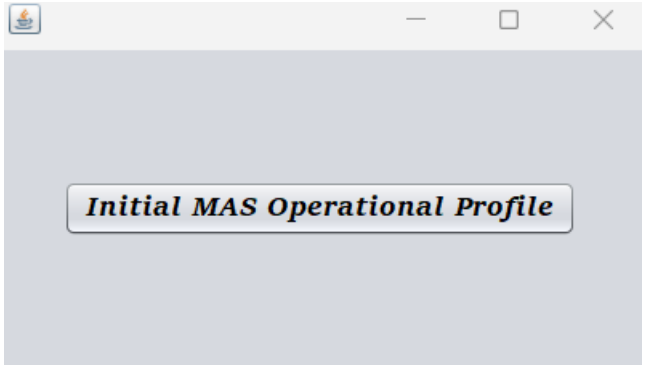

Figure 11. Interface d'accueil de l'outil de génération.

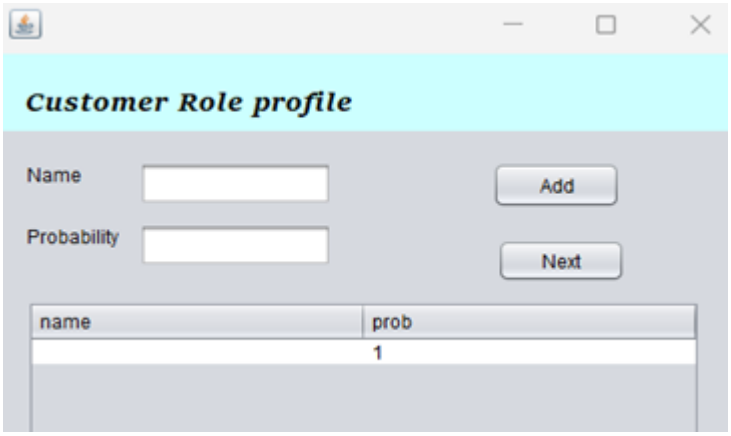

Figure 12. Interface du profil de rôle client.

2.3. Étape 2 : Profil de rôle utilisateur (User Role Profile)

Un profil de rôle utilisateur n'est pas nécessairement identique à un profil de rôle client. Un utilisateur peut être un agent ou un acteur, un groupe d'agents ou d'acteurs qui utilise le système. Un groupe de rôles d'utilisateurs est un ensemble de rôles d'utilisateurs qui utiliseront le système de la même manière. Le profil de rôle d'utilisateur correspond aux groupes de rôles d'utilisateur, qui utilisent le système différemment, associés à leur probabilité d'occurrence.

Le profil de rôle utilisateur doit être créé après avoir défini le profil de rôle client. S'il existe plusieurs groupes de rôles de clients, les probabilités des groupes de rôles d'utilisateurs doivent être multipliées par les probabilités des groupes de rôles de clients pour obtenir leurs probabilités globales.

Le tableau 3 montre le profil du rôle de l'utilisateur du système universitaire. Nous avons identifié cinq types de rôle d'utilisateur : Enseignant-Chercheur (Teacher-researcher), Etudiant (Student), Chercheur (Researcher), Porteur de projet (Project leader) et Administration. De plus, supposant que leurs probabilités d'occurrence sont de 0,13, 0,7, 0,06, 0,01 et 0,1 respectivement. Notez que les probabilités globales de groupe de rôles d'utilisateurs sont les mêmes que les probabilités d'occurrence de groupes de rôles d'utilisateurs car les probabilités de groupes de rôles de clients sont de 1 dans notre cas, comme il est expliqué précédemment.

Tableau 3. User role profile of university system [76].

User role	Overall user role group probability
Teacher-Researcher	0.13
Student	0.7
Researcher	0.06
Project leader	0.01
Administration	0.1

La figure 13 présente l'interface qui permet de déterminer les utilisateurs de système et leurs probabilités ainsi que la figure 14 montre le profil de rôle utilisateur.

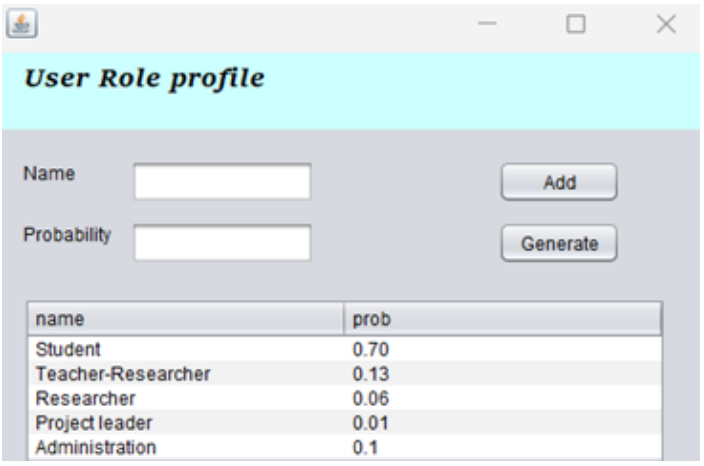

Figure13. Interface de rôles d'utilisateur.

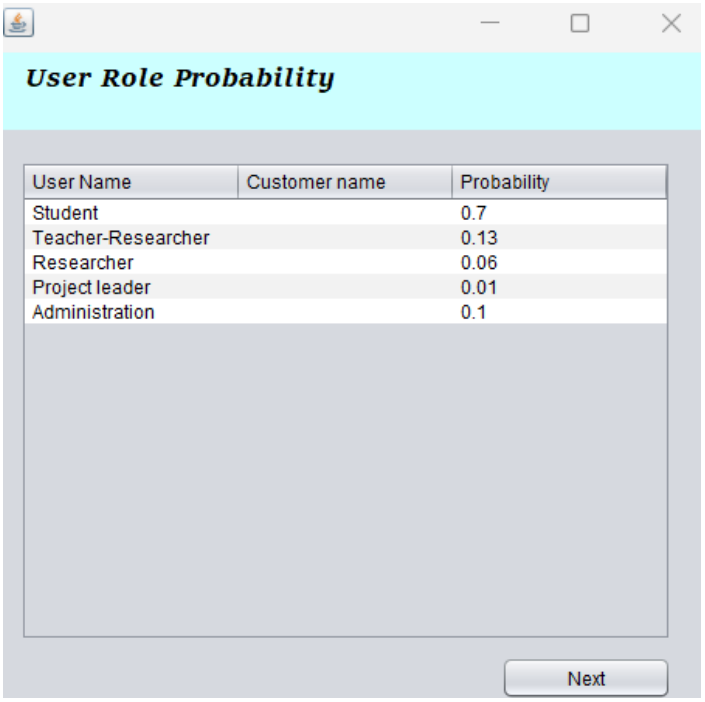

Figure 14. Interface du profil de rôle utilisateur.

2.4. Étape 3 : Profil de mode (Mode profile)

Un mode NorMAS est défini comme un ensemble d'objectifs et de sous-objectifs regroupés pour déterminer l'environnement d'exécution. Pour chaque mode système, un profil opérationnel doit être déterminé.

Un ensemble de modes système et leur probabilité d'occurrence sont appelés profil de mode NorMAS. On distingue deux modes dans le système universitaire : le mode cours (Course mode) et mode de recherche (Research mode). Nous supposons que le mode cours utilise 80% du rôle d'enseignant-chercheur, 100% du rôle d'étudiant et 50% du rôle d'administration. Le reste des pourcentages d'enseignant-chercheur et d'administration (20% et 50%) avec 100% de rôles de chercheur et de porteur de projet sont utilisés par le mode recherche.

La probabilité globale du mode cours est calculée en multipliant les probabilités d'occurrence des groupes de rôles d'utilisateurs (enseignant-chercheur, 0,13 ; étudiant, 0,7 ; et administration, 0,1) par la proportion d'utilisation (0,8 ; 1 ; et 0,5) respectivement et en additionnant les résultats. En utilisant la même formule, nous calculons la probabilité globale du mode de recherche. Le profil de mode système dérivé est présenté dans le Tableau 4.

Tableau 4. System mode profile of university system [76].

System mode	Occurrence Probability
Course Mode	0.854
Research Mode	0.146

L'interface qui nous permet d'initier les modes de système est montrée par la figure 15. La figure 16 représente l'initialisation des probabilités des rôles d'utilisateur pour chaque mode de système. Le résultat est affiché dans la figure 17.

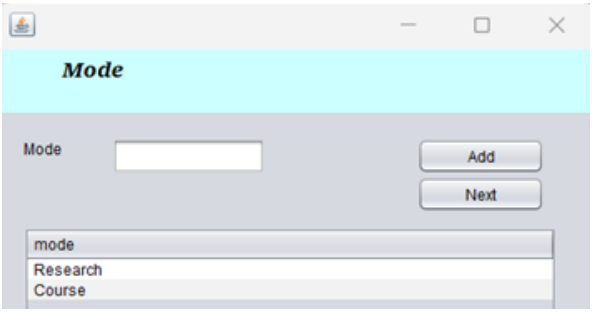

Figure 15. Interface de modes de système.

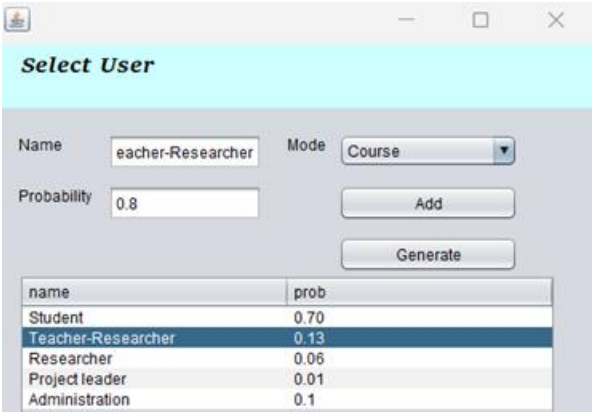

Figure 16. Interface de probabilités des utilisateurs par mode.

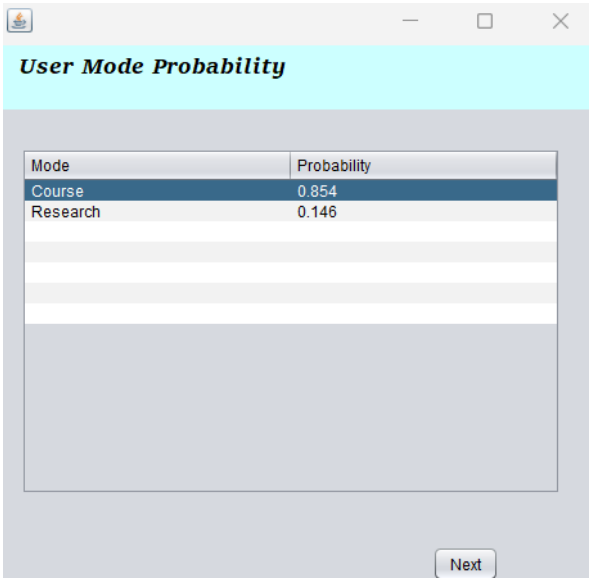

Figure 17. Interface du profil de mode système.

2.5. Étape 4 : Profil d'objectif (Goal profile)

Comme son nom l'indique, le profil d'objectifs est un ensemble d'objectifs avec leur probabilité d'occurrence. Il doit être développé pour chaque mode du système et ses objectifs sont définis au niveau de la conception. Pour développer le profil d'objectif, il est nécessaire de trouver la liste initiale des objectifs qui décrivent l'objectif principal du système et les variables environnementales qui peuvent affecter les objectifs du système lors de l'exécution. Ensuite, la liste d'objectifs initiale et les variables environnementales sont combinées pour déterminer la liste d'objectifs finale. Par manque d'espace, nous expliquerons les deux dernières étapes de la méthodologie proposée à travers un seul mode, qui est le mode cours.

Liste d'objectifs initiale. Dans cette étape, nous devons identifier la liste des objectifs de base de l'agent du système. Nous pouvons l'obtenir à partir de la spécification du système décrite dans la section 3.1. La liste des objectifs initiaux du mode cours est indiquée dans le tableau 5.

Tableau 5. Liste d'objectifs initiale du mode cours [76].

Goals	Agent role	Course Mode (0.854)	
		Occurrence probability	Overall occurrence probability
Goal1 (Teach)	Teacher-Researcher	0.2	0.1708
Goal2 (Evaluate)	Teacher-Researcher	0.1	0.0854
Goal4 (Respect regulations)	Teacher-Researcher, Student	0.1	0.0854
Goal5 (Occupy a senior position)	Teacher-Researcher	0.05	0.0427
Goal6 (Guided & Autonomous practice)	Student	0.2	0.1708
Goal7 (Pass evaluation)	Student	0.1	0.0854
Goal8 (Attend courses)	Student	0.1	0.0854
Goal16(Punish)	Administration	0.075	0.06405
Goal17(recompense)	Administration	0.075	0.06405

La figure 18 montre comment entrer les objectifs de chaque rôle d'utilisateur et leurs probabilités afin de calculer les probabilités globales des objectifs pour chaque mode de système.

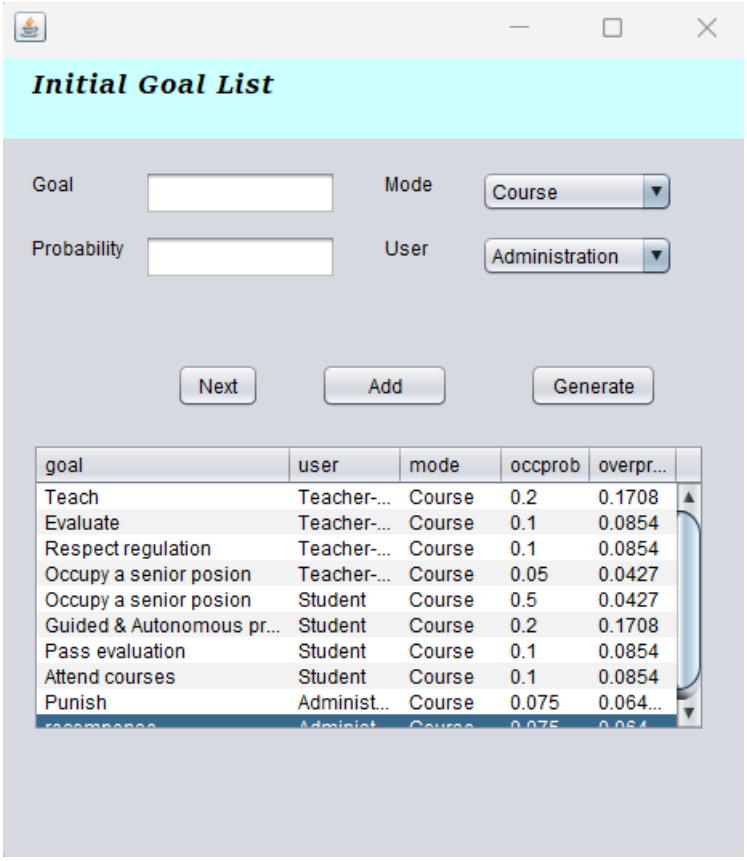

Figure 18. Interface de la liste d'objectifs initiale.

Tableau 6. Variables d'environnement de système[76].

NorMAS Goals	Environmental Variables	
	NorMAS agent and actor variables	NorMAS variables
Goal1 (Teach)	Absence of student role (Abs-SR)	NorMAS Resources Problems (NorMAS-RP)
Goal2 (Evaluate)		
Goal3 (Research)	Absence of project leader role (Abs-LR)	
Goal4 (Respect regulations)	-	
Goal5 (Occupy a senior position)		
Goal6 (Guided & Autonomous practice)	Absence of teacher-researcher role (Abs-TR)	
Goal7 (Pass evaluation)		
Goal8 (Attend courses)		
Goal9 (Collaborate with researchers)	-	
Goal10 (Advance in research)	Absence of project leader role (Abs-LR)	
Goal11 (Attend seminars)	-	
Goal12 (Make industrial contacts)		
Goal13 (Sign contracts)		
Goal14 (Propose approach)		
Goal15 (Supervising researcher)	Absence of Researcher role (Abs-RR)	
Goal16(Punish)	Absence of student and teacher-researcher roles (Abs-SR&TR)	
Goal17(recompense)		

Variables d'environnement. Les variables d'environnement sont les événements qui peuvent affecter l'exécution du système. Selon les spécificités des systèmes multi-agents, il faut considérer deux types de variables environnementales : les variables liées au système lui-même et celles liées aux différents agents

et acteurs du système. Dans notre cas, nous considérons les variables environnementales présentées dans le tableau 6. Le problème des ressources didactiques, pédagogiques et matérielles (NorMAS-RP) est considéré comme une variable liée au système. Pour les différents agents du système, on considère l'absence de rôle variable (absence de rôle d'étudiant, absence de rôle de chef de projet...).

Liste finale des objectifs. Après avoir pris en compte les facteurs environnementaux, nous pouvons calculer la liste d'objectifs finale en multipliant la liste d'objectifs initiale et les variables d'environnement. Le tableau 7 montre le profil d'objectif final du mode cours.

Tableau 7. Profil d'objectif du mode cours [76].

Goals	Norm type	Agent role	Course Mode (0.854)	
			Environmental variables	Occurrence probability
Goal1 (Teach)	Obligation	Teacher-Researcher	NorMAS-RP (0.01)	0.001708
			Abs-SR (0.01)	0.001708
			Normal conditions (0.98)	0.167384
Goal2 (Evaluate)			NorMAS-RP (0.01)	0.000854
			Abs-SR (0.01)	0.000854
			Normal conditions (0.98)	0.083692
Goal4 (Respect regulations)		Teacher-Researcher, Student	NorMAS-RP (0.01)	0.000854
			Normal conditions (0.99)	0.084546
Goal5 (Occupy a senior position)	Permission	Teacher-Researcher	NorMAS-RP (0.01)	0.000427
			Normal conditions (0.99)	0.042273
Goal6 (Guided & Autonomous practice)	Obligation	Student	NorMAS-RP (0.01)	0.001708
			Abs-TR (0.01)	0.001708
			Normal conditions (0.98)	0.167384
Goal7 (Pass evaluation)	Obligation	Student	NorMAS-RP (0.01)	0.000854
			Abs-TR (0.01)	0.000854
			Normal conditions (0.98)	0.083692
Goal8 (Attend courses)	Recommendation		NorMAS-RP (0.01)	0.000854
			Abs-TR (0.01)	0.000854
			Normal conditions (0.98)	0.083692
Goal16 (Punish)	Permission	Administration	NorMAS-RP (0.01)	0.0006405
			Abs-SR&TR (0.001)	0.00006405
			Normal conditions (0.989)	0.06334545
Goal17 (recompense)			NorMAS-RP (0.01)	0.0006405
			Abs-SR&TR (0.001)	0.00006405
			Normal conditions (0.989)	0.06334545

2.6. Étape 5 : Profil Opérationnel NorMAS (NorMAS Operational Profile)

Le profil opérationnel NorMAS est un ensemble d'opérations avec leur probabilité d'occurrence. Chaque objectif de chaque mode système se décompose en un ensemble de sous-objectifs, eux-mêmes décomposés en un ensemble d'actions (opérations). Par exemple, le scénario d'enseignement (objectif 1), présenté par un diagramme d'activités (figure 20), comprend quatre actions : préparer le cours, au cours duquel l'enseignant-chercheur prépare sa présentation ; cours présent, au cours duquel l'enseignant-chercheur démontre les apprentissages à accomplir ; pratique guidée, au cours de laquelle l'enseignant-chercheur,

avec les étudiants, réalise des tâches similaires à celles réalisées lors de la présentation du cours ; et préparer une pratique autonome, au cours de laquelle l'enseignant-chercheur prépare un travail personnel et laisse l'étudiant réinvestir seul ce qu'il a appris lors des deux étapes précédentes. La probabilité d'occurrence globale de chaque action est calculée en multipliant la probabilité d'action initiale par la probabilité d'occurrence de son objectif. Si l'objectif ne comprend qu'une seule action, la probabilité d'occurrence globale de cette action est la même que la probabilité d'occurrence globale de son objectif. Le profil opérationnel du mode course sera présenté dans le tableau 8.

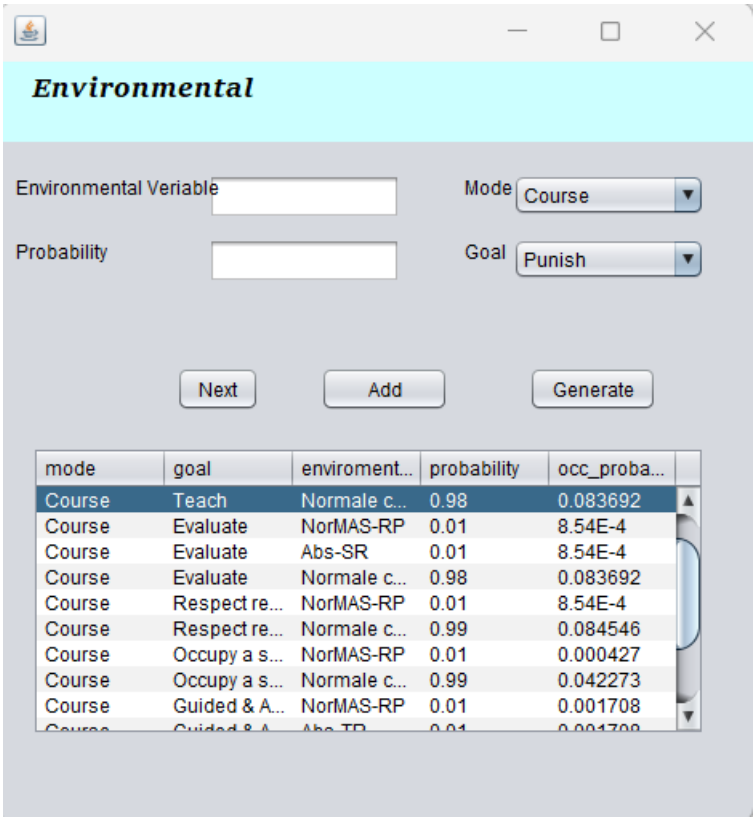

Figure 19. Interface du profil d'objectif du mode cours.

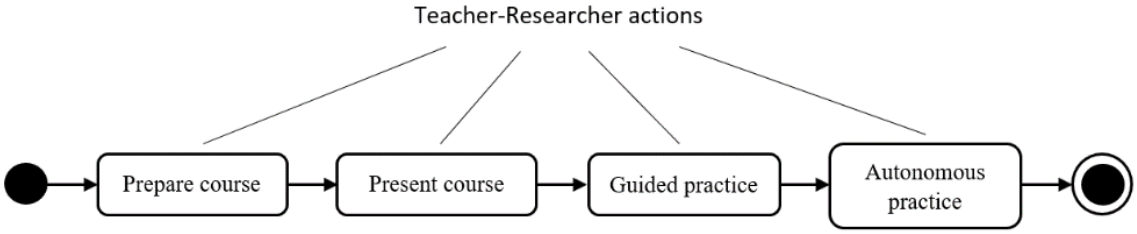

Figure 20. Scénario de l'objectif « Enseigner » [76].

Tableau 8. Profil opérationnel du mode cours [76].

Agent role	Actions (operations)	Course Mode (0.854)		
		Occurrence probability	Environmental variables	Overall occurrence probability
Teacher-Researcher	Goal1 (Teach)			
	Prepare course	0.2	NorMAS-RP (0.01)	0.0003416
			Abs-SR (0.01)	0.0003416
			Normal conditions (0.98)	0.0334768
	Present course	0.3	NorMAS-RP (0.01)	0.0005124
			Abs-SR (0.01)	0.0005124

			Normal conditions (0.98)	0.0502152
	Guided practice	0.3	NorMAS-RP (0.01)	0.0005124
			Abs-SR (0.01)	0.0005124
			Normal conditions (0.98)	0.0502152
	Prepare autonomous practice	0.2	NorMAS-RP (0.01)	0.0003416
			Abs-SR (0.01)	0.0003416
			Normal conditions (0.98)	0.0334768
Teacher-Researcher	Goal2 (Evaluate)			
	Prepare the evaluation work	0.2	NorMAS-RP (0.01)	0.0001708
			Abs-SR (0.01)	0.0001708
			Normal conditions (0.98)	0.0167384
	Prepare typical answer	0.2	NorMAS-RP (0.01)	0.0001708
			Abs-SR (0.01)	0.0001708
			Normal conditions (0.98)	0.0167384
	Correct evaluation work	0.5	NorMAS-RP (0.01)	0.000427
			Abs-SR (0.01)	0.000427
			Normal conditions (0.98)	0.041846
	Assign marks	0.1	NorMAS-RP (0.01)	0.0000854
			Abs-SR (0.01)	0.0000854
			Normal conditions (0.98)	0.0083692
Teacher-Researcher, Student	Goal4 (Respect regulations)			
	Know the regulations	0.2	NorMAS-RP (0.01)	0.0001708
			Normal conditions (0.99)	0.0169092
	Practice rights	0.4	NorMAS-RP (0.01)	0.0003416
			Normal conditions (0.99)	0.0338184
	Apply duties	0.4	NorMAS-RP (0.01)	0.0003416
			Normal conditions (0.99)	0.0338184
Teacher-Researcher	Goal5 (Occupy a senior position)			
	Occupy a senior position	1	NorMAS-RP (0.01)	0.000427
			Normal conditions (0.99)	0.042273
Student	Goal6 (Guided & Autonomous practice)			
	Achieve guided practice	0.6	NorMAS-RP (0.01)	0.0010248
			Abs-TR (0.01)	0.0010248
			Normal conditions (0.98)	0.1004304
	Achieve autonomous practice	0.4	NorMAS-RP (0.01)	0.0006832
			Abs-TR (0.01)	0.0006832
			Normal conditions (0.98)	0.0669536
	Goal7 (Pass evaluation)			
	Responds to evaluation work	1	NorMAS-RP (0.01)	0.000854
			Abs-TR (0.01)	0.000854
			Normal conditions (0.98)	0.083692
	Goal8 (Attend course)			
	Attend course presentation	1	NorMAS-RP (0.01)	0.000854
			Abs-TR (0.01)	0.000854
			Normal conditions (0.98)	0.083692
Administration	Goal16 (Punish)			
	Apply punishment	1	NorMAS-RP (0.01)	0.0006405
			Abs-SR&TR (0.001)	0.00006405
			Normal conditions (0.989)	0.06334545
	Goal17 (recompense)			
	Offer a recompense	1	NorMAS-RP (0.01)	0.0006405
			Abs-SR&TR (0.001)	0.00006405
			Normal conditions (0.989)	0.06334545

L'interface qui présente le profil opérationnel final du mode cours de notre système « Université » est montrée par la figure 21.

3. Langages utilisés dans le développement de notre outil

Afin de valider l'approche proposée [76], nous avons développé un outil pour la génération automatique du profil opérationnel pour les SMA normatifs. Cet outil est implémenté essentiellement grâce aux langages

Java, SQL et l'IDE NetBeans. Un aperçu sur les langages utilisés pour le développement de notre outil est présenté ci-dessous.

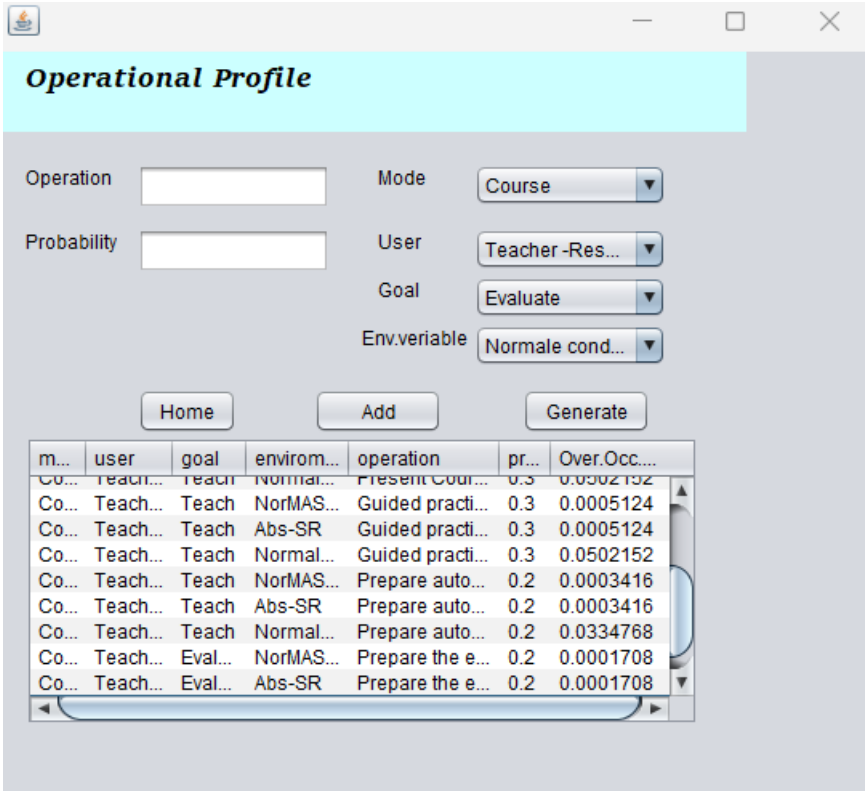

Figure 21. Interface du profil opérationnel final généré.

3.1. Langage Java

Java [86] est à la fois un langage de programmation et un environnement d'exécution. Le langage Java a la particularité principale d'être portable sur plusieurs systèmes d'exploitation tels qu'UNIX, Microsoft Windows, Mac OS ou Linux. C'est la plateforme qui garantit la portabilité des applications développées en Java. Le langage reprend en grande partie la syntaxe du langage C++, très utilisé par les informaticiens. Néanmoins, Java a été épurée des concepts les plus subtils du C++ et à la fois les plus déroutants, tels que l'héritage multiple remplacé par l'implémentation des interfaces. Les concepteurs ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.).

3.2. L'IDE NetBeans

NetBeans IDE [87] est un environnement de développement intégré gratuit et à code source ouvert destiné au développement d'applications sous Windows, Mac, Linux et Solaris.

L'environnement IDE simplifie le développement d'applications Web, d'entreprise, de bureau et mobiles utilisant les plates-formes Java et HTML5. Il offre également une assistance pour le développement d'applications PHP et C/C++.

3.3. SQL

SQL [88] est un langage de programmation permettant de manipuler les données et les systèmes de bases de données relationnelles. Ce langage permet principalement de communiquer avec les bases de données

afin de gérer les données qu'elles contiennent. Il permet notamment de stocker, de manipuler et de retrouver ces données. Il est aussi possible d'effectuer des requêtes, de mettre à jour les données, de les réorganiser, ou encore de créer et de modifier le schéma et la structure d'un système de base de données et de contrôler l'accès à ses données.

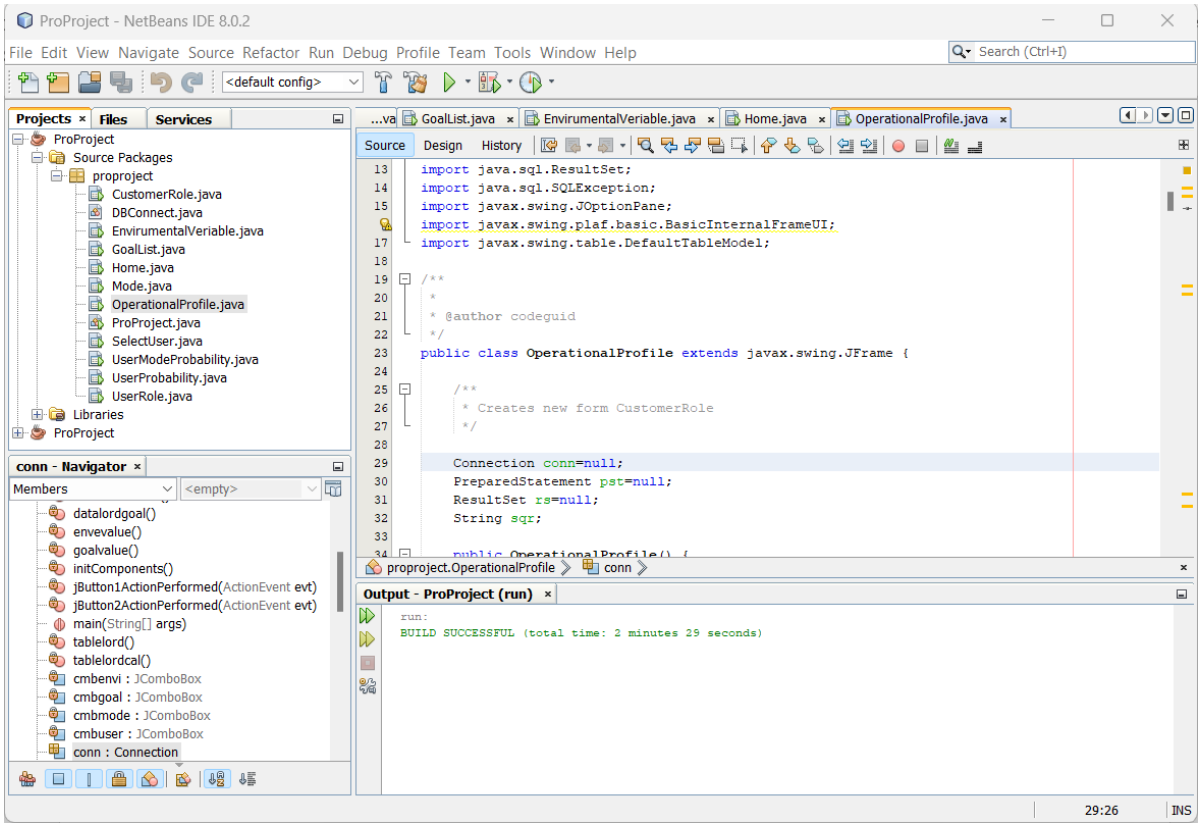

Figure 22. NetBeans IDE and java codes.

3.4. Wampp Server

WampServer [89] est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PHPMyAdmin pour gérer plus facilement vos bases de données.

Conclusion

Afin de tester la fiabilité des systèmes multi-agents normatifs (NorMAS), ce chapitre a introduit, dans sa première partie, une nouvelle approche de test des SMA normatifs. Cette approche est basée sur une récente méthodologie de développement du profil opérationnel pour ce type de systèmes [76]. Dans la seconde partie, nous avons développé un outil afin d'automatiser la méthodologie proposée dans [76] qui est illustré à travers une étude de cas concrète d'un système multi-agent normatif appelé « Université ». A la fin du chapitre, nous avons présenté les langages utilisés lors de l'implémentation de notre outil. Dans ce qui suit, nous concluons notre manuscrit.

Conclusion générale & Perspectives

" Science sans conscience n'est que ruine de l'âme."

François Rabelais

1. Conclusion générale

De nos jours, les tests représentent le principal moyen de validation du bon fonctionnement d'un logiciel. Leur but est d'analyser ou d'exécuter un programme afin de détecter un maximum de défauts et de mettre en évidence les points où le comportement n'est pas conforme aux attentes. Les tests restent un moyen efficace pour évaluer la fiabilité d'un logiciel ou d'un composant logiciel. L'activité de test est présente tout au long du cycle de vie du logiciel et utilise différentes techniques pour valider les différentes étapes du développement.

Dans ce mémoire, nous avons concentré notre étude sur les systèmes multi-agents (SMA) normatifs, et plus particulièrement sur leur test de fiabilité. Par conséquent, nous avons introduire une nouvelle approche de test de fiabilité des SMA normatifs. Cette approche est basée sur une méthodologie de développement du profil opérationnel (PO) pour ce type de systèmes, proposée récemment [76], ce qui nous a conduit à automatiser le processus de développement de cette méthodologie. Afin d'illustrer le processus d'automatisation, nous avons présenté un exemple concret d'un SMA normatif (Université) comme une étude de cas.

2. Perspectives

Notre travail ne s'arrête pas à ce stade, en effet plusieurs fonctionnalités peuvent être ajoutées à notre outil développé pour la génération du PO pour les SMA normatifs. A savoir, impliquer les normes (Obligations, Interdiction, Permissions et Recommandations) dans le processus de génération du PO.

Références bibliographiques

[1] Mokhati, F. (2021). Systèmes Multi-Agents. Support de cours. Université d'Oum el bouaghi.

[2] Demazeau, Y. V. (1996). 1st Iberoamerican Workshop on Distributed Artificial Intelligence and Multi-Agent Systems, IWDAIMAS'96.

[3] Ferber, J. (1995). Les Systèmes Multi Agents : vers une intelligence collective. InterEditons.

[4] Wooldridge, M. (2009). An Introduction to MultiAgent Systems. Jhon wiely and sons, Ltd, Publication.

[5] Marir, T. (2017). Les Systèmes Multi-Agents. Cours, master2, architecture distribuée. Université d'Oum el bouaghi.

[6] Stratulat, T. (2004). HAL thèse. Récupéré sur archives-ouvertes : https://tel.archives-ouvertes.fr/tel-00005215v2

[7] Weyns, D., Omicini, A., & Odell, J. (2007). Environment as a first-class abstraction in multiagent.

[8] Russell, S., & Norvig, P. (2010). Artificial Intelligence: A Modern Approach. New Jersey: Pearson Education.

[9] Larousse, norme. (2022). Récupéré sur larousse : https://www.larousse.fr/dictionnaires/francais/norme/55009

[10] Merriam-webster, Norm. (2022). Récupéré sur merriam-webster: https://www.merriam-webster.com/dictionary/norm

[11] Mahmoud, M., Ahmad, M., & Mohd Yusoff, M. (2014). A Review of Norms and Normative Multiagent Systems. The Scientific World Journal.

[12] Boella, G., van der Torre, L., & Verhagen, H. (2006). Introduction to Normative Multiagent Systems. Computational & Mathematical Organization Theory.

[13] Boella, G., & van der Torre, L. (2008). Substantive and procedural norms in normative multiagent systems. Journal of Applied Logic.

[14] Hollander, C., & Annie, S. (2011). The Current State of Normative Agent-Based Systems. Artificial Societies and Social Simulation.

[15] Abar, S., Theodoropoulos, G., Lemarinier, P., & O'Hared, G. (2017). Agent Based Modelling and Simulation tools: A review of the state-of-art software. Computer Science Review.

[16] Allan, R. (2010). Survey of Agent Based Modelling and Simulation Tools. Warrington: Science and Technology Facilities Council.

[17] R. Pawlak, et al., "Programmation orientée aspect pour Java / J2EE", édition Eyrolles, 2004.

[18] G. J. Myers, C. Sandler, T. Badgett, and T. M. Thomas, "The Art of Software Testing", Second Edition, Wiley, 2004.

[19] T. Gil, "Conception orientée aspect, version 2.1", DotnetGuru, 2007, http://www.lulu.com/content/8309554.

[20] L' AFCIQ, Agence Française de Contrôle Industriel de la Qualité.

[21] F. Duclos, "Environnement de Gestion de Services Non Fonctionnels Dans les Applications à Composants", Doctoral dissertation, Université Joseph Fourier, Octobre 2002.

[22] Xanthakis S, Régnier P and Karapoulios C, Le test des logiciels, Hermes sciences publications, 2000.

[23] AlI GUECHI, Farida (2014), Une approche de test des systemes multiagents basee sur des modeles formwls. Magister thesis, Université de Batna 2.

[24] Department of Defense, "Military Standard Defense System Software Development," February 1988.

[25] Glossaire CFTL/ISTQB des termes utilisés en tests de logiciels Version 1.0FR Traduction française de la Version 1.0 produite par 'Glossary Working Party' International Software Testing Qualification Board en date du (08 Décembre, 2004).

[26] Présentation au sujet: " Test de logiciel GLG101 AP.TELLE & S.MILOVANOVIC MAI 2007 ", https://slideplayer.fr/slide/517335/.

[27] Hamlet R, Test du logiciel & confiance, Génie logiciel et systèmes experts, 30, mars 1993.

[28] Low, C.K., Chen, T.Y. & Rónnquist, R. Automated Test Case Generation for BDI Agents. *Autonomous Agents and Multi-Agent Systems* **2**, 311–332 (1999). https://doi.org/10.1023/A:1010011219782.

[29] Zina Houhamdi, Multi-Agent System Testing: A Survey, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 6, 2011.

[30] Huget MP and Demazeau Y, Evaluating multiagent systems: a record/replay approach. Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on 2004, pp. 536-539.

[31] Nguyen, C.D.: Testing Techniques for Software Agents, PhD thesis, International Doctorate School in Information and Communication Technologies - University of Trento (2008).

[32] Moreno, M., Pav´on, J., Rosete, A.: Testing in agent oriented methodologies. In: Omatu, S., Rocha, M.P., Bravo, J., Fern´andez, F., Corchado, E., Bustillo, A., Corchado, J.M. (eds.) IWANN 2009. LNCS, vol. 5518, pp. 138–145. Springer, Heidelberg (2009).

[34] Zhang, Z., Thangarajah, J., Padgham, L.: Automated unit testing for agent systems. In: 2nd International Working Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2007), Barcelona, Spain, pp. 10–18 (2007).

[35] Zhang, Z., Thangarajah, J., Padgham, L.: Automated unit testing intelligent agents in pdt. In: AAMAS (Demos), pp. 1673–1674 (2008).

[36] Zhang, Z., Thangarajah, J., Padgham, L.: Model based testing for agent systems. In: AAMAS, vol. (2), pp. 1333–1334 (2009).

[37] Padgham, L., Winikoff, M.: Developing Intelligent Agent Systems: A Practical Guide. John Wiley and Sons, Chichester (2004).

[38] Ekinci, E.E., Tiryaki, A.M., Cetin, O., Dikenelli, O.: Goal-Oriented Agent Testing Revisited. In: Proc. of the 9th Int. Workshop on Agent-Oriented Software Engineering, pp. 85–96 (2008).

[39]Lam, D.N., Barber, K.S.: Debugging agent behavior in an implemented agent system. In: Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2004. LNCS (LNAI), vol. 3346, pp. 104–125. Springer, Heidelberg (2005).

[40] Nunez, M., Rodréguez, I., Rubio, F.: Specification and testing of autonomous agents in ecommerce systems. Software Testing, Verification and Reliability 15(4), 211–233 (2005).

[41] Coelho, R., Kulesza, U., von Staa, A., Lucena, C.: Unit testing in multi-agent systems using mock agents and aspects. In: SELMAS 2006: Proceedings of the 2006 International Workshop on Software Engineering for Large-scale Multi-agent Systems, pp. 83–90. ACM Press, New York (2006), http://dx.doi.org/http://doi.acm.org/10.1145/1138063.1138079.

[42] Gamma, E., Beck, K.: JUnit: A Regression Testing Framework (2000), http://www.junit.org.

[43] Nguyen, C.D., Perini, A., Tonella, P.: Ontology-based Test Generation for Multi Agent Systems. In: Proc. of the International Conference on Autonomous Agents and Multiagent Systems (2008).

[44] Nguyen, C.D., Miles, S., Perini, A., Tonella, P., Harman, M., Luck, M.: Evolutionary testing of autonomous software agents. In: The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), pp. 521–528. IFAAMAS (2009).

[45] Serrano, E., Botia, J.A.: Infrastructure for forensic analysis of multi-agent systems. In: Hindriks, K.V., Pokahr, A., Sardina, S. (eds.) ProMAS 2008. LNCS, vol. 5442, pp. 168–183. Springer, Heidelberg (2009).

[46] Serrano, E., Gòmez-Sanz, J.J., Botia, J.A., Pavon, J.: Intelligent data analysis applied to debug complex software systems. Neurocomputing 72(13-15), 2785– 2795 (2009).

[47] Tiryaki, A.M., Öztuna, S., Dikenelli, O., Erdur, R.C.: SUNIT: A unit testing framework for test driven development of multi-agent systems. In: Padgham, L., Zambonelli, F. (eds.) AOSE VII / AOSE 2006. LNCS, vol. 4405, pp. 156–173. Springer, Heidelberg (2007).

[48] Dikenelli, O., Erdur, R.C., Gumus, O.: Seagent: a platform for developing semantic web based multi agent systems. In: AAMAS 2005: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1271–1272. ACM Press, New York (2005), http://dx.doi.org/http://doi.acm.org/10.1145/1082473.1082728.

[49] Gomez-Sanz, J.J., Bot´a, J., Serrano, E., Pav´on, J.: Testing and debugging of MAS interactions with INGENIAS. In: Luck, M., Gomez-Sanz, J.J. (eds.) AOSE 2008. LNCS, vol. 5386, pp. 199–212. Springer, Heidelberg (2009).

[50] Nguyen, C.D., Perini, A., Tonella, P.: Goal-oriented testing for MASs. Int. J.Agent-Oriented Software Engineering 4(1), 79–109 (2010).

[51] ANSI/IEEE, Standard Glossary of Software Engineering Terminology, STD-729-1991, ANSI/IEEE, 1991.

[52] Joy Anderson, Le test de fiabilité dans le test logiciel | Guide complet. Dans : Automatisation des tests (11 Octobre 2022), https://fr.agilitest.com/blog/reliability-testing-software-testing-complete-guide.

[53] M.L. Shooman, Reliability of Computer Systems and Networks: Fault Tolerance, Analysis and Design, Wiley, New York, 2002.

[54] M. Chen, M.R. Lyu, and E. Wong, "Effect of Code Coverage on Software Reliability Measurement," IEEE Transactions on Reliability, vol. 50, no. 2, June 2001, pp.165-170.

[55] Y.K. Malaiya, N. Li, J.M. Bieman, and R. Karcich, "Software Reliability Growth with Test Coverage," IEEE Transactions on Reliability, vol. 51, no. 4, December 2002, pp. 420-426.

[56] M.A. Vouk, "Using Reliability Models During Testing With Nonoperational Profiles," in Proceedings of 2nd Bellcore/Purdue Workshop on Issues in Software Reliability Estimation, October 1992, pp. 103-111.

[57] X. Teng, H. Pham, and D. Jeske, "Reliability Modeling of Hardware and Software Interactions, and Its Applications," IEEE Transactions on Reliability, vol. 55, no. 4, Dec. 2006, pp. 571-577.

[58] A. von Mayrhauser and D. Chen, "Effect of Fault Distribution and Execution Patterns on Fault Exposure in Software: A Simulation Study," Software Testing, Verification & Reliability, vol. 10, no.1, March 2000, pp. 47- 64.

[59] M.R. Lyu (ed.), Handbook of Software Reliability Engineering, IEEE Computer Society Press and McGraw-Hill, 1996.

[60] J.D. Musa, "Operational Profiles in Software Reliability Engineering," IEEE Software, Volume 10, Issue 2, March 1993, pp. 14-32.

[61] Musa J-D (1992) The operational profile in software reliability engineering: an overview. [1992] Proceedings Third International Symposium on Software Reliability Engineering, Research Triangle Park, NC, USA, 1992, 140-154.

[62] Juhlin B-D (1992) Implementing operational profiles to measure system reliability. In: Proceedings of the 3rd International Symposium on Software Reliability Engineering (ISSRE'02), Research Triangle Park, NC, USA, 286–295.

[63] Pant H, Franklin P, Everett W (1994) A structured approach to improving software-reliability using operational profiles. Proceedings of Annual Reliability and Maintainability Symposium (RAMS), Anaheim, CA, USA, 142-146.

[64] Elbaum S, Narla S (2001) A methodology for operational profile refinement. Annual Reliability and Maintainability Symposium. 2001 Proceedings. International Symposium on Product Quality and Integrity (Cat. No.01CH37179), Philadelphia, PA, USA, 142-149.

[65] Gittens M, Lutfiyya H, Bauer M (2004) An extended operational profile model. 15th. International Symposium on Software Reliability Engineering, Saint-Malo, France, 314-325.

[66] Shukla R, Carrington D, Strooper P-A (2004) Systematic operational profile development for software components. 11th. Asia-Pacific Software Engineering Conference, Busan, Korea (South), 528-537.

[67] Arora S, Misra R-B (2005) Software reliability improvement through operational profile testing. Annual Reliability and Maintainability Symposium, 2005. Proceedings. Alexandria, VA, USA, 621-627.

[68] Yamany H-E and Capretz M-A-M (2007) A Multi-Agent Framework for Building an Automatic Operational Profile. In: Elleithy K. (eds) Advances and Innovations in Systems, Computing Sciences and Software Engineering, Springer, Dordrecht, 161-166.

[69] Kumar K-S, Misra R-B (2007) Software operational profile-based test case allocation using fuzzy logic. International Journal of Automation and Computing, Vol. 04, No. 4, 388–395.

[70] Fu J-P, Lu M-Y (2009) Develop software operational profile with uniform design. 2009 IEEE International Conference on Industrial Engineering and Engineering Management, Hong Kong, China, 842-846.

[71] Muhammad Ali-Shahid M, Sulaiman S (2014) A review of software operational profile in software reliability engineering. 2014 8th. Malaysian Software Engineering Conference (MySEC), Langkawi, Malaysia, 1-6.

[72] Smidts C, Mutha C, Rodriguez M, Gerber M-J (2014) Software Testing with an Operational Profile: OP Definition. In: journal ACM Computing Surveys (CSUR), Vol. 46, No. 3, Article 39, 1-39.

[73] Amrita, Yadav D-K (2017) Development of Software Operational Profile. International Journal of Applied Engineering Research, Vol. 12, No. 22, 11865-11873.

[74] Cavamura L (2019) Operational Profile and Software Testing: Aligning User Interest and Test Strategy. 2019 12th. IEEE Conference on Software Testing, Validation and Verification (ICST), Xi'an, China, 492-494.

[75] Fazzolino R, Rodrigues G-N (2019) Feature-Trace: an approach to generate operational profile and to support regression testing from BDD features. Proceedings of the XXXIII Brazilian Symposium on Software Engineering (SBES 2019), 332-336.

[76] Menassel, Y., Marir, T., Mokhati, F (2023) An Operational Profile for Normative Multi-agent Systems. In: Gupta, V., Rubalcaba, L., Gupta, C., Hanne, T. (eds) Sustainability in Software Engineering and Business Information Management. SSEBIM 2022. Lecture Notes in Information Systems and Organisation, vol 62. Springer, Cham.

[77] M. R. Lyu (2007) Software Reliability Engineering: A Roadmap. Future of Software Engineering (FOSE), Minneapolis, MN, pp. 153-170.

[78] S. U. Farooq, S. Quadri and N. Ahmad (2012) Metrics, models and measurements in software reliability, IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 441-449.

[79] C. Kaner (2003) "What Is a Good Test Case?"; What Is a Good Test Case?, pp. 1-13.

[80] C. Liu, Z. Ren, H. Li and Y. Liu (2014) Software reliability testing in practice: An industry case study with a typical airborne software system, 10th International Conference on Reliability, Maintainability and Safety (ICRMS), Guangzhou, pp. 363-368.

[81] K. Sharma, R. Garg, C. K. Nagpal and R. K. Garg (2010) Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach, in IEEE Transactions on Reliability, vol. 59, no. 2, pp. 266-276.

[82] N. Ullah, M. Morisio and A. Vetro (2012) A Comparative Analysis of Software Reliability Growth Models using Defects Data of Closed and Open-Source Software, 35th Annual IEEE Software Engineering Workshop, Heraclion, pp. 187- 192.

[83] G. Gayathry and R. Thirumalai Selvi (2016) Classification of Software Reliability Models to Improve the Reliability of Software", Indian Journal of Science and Technology, vol 8, pp. 1-5.

[84] A. Kumar (2016) Software Reliability Growth Models, Tools and Data Sets- A Review, ISEC '16: Proceedings of the 9th India Software Engineering Conference, Goa, pp. 80-88.

[85] T. Marir, F. Mokhati, H. Bouchlaghem-Seridi, Y. Acid, and M. Bouzid (2016) QM4MAS: a quality model for multi-agent systems," International Journal of Computer Applications in Technology (IJCAT), vol. 54, no. 4.

[86] Site officiel Java, http://www.java.com/fr/download/faq/whatis_java.xml.

[87] Site officiel NetBeans, https://www.oracle.com/dz/tools/technologies/netbeans-ide.html.

[88] SQL : Tout savoir sur le langage des bases de données, https://datascientest.com/sql-tout-savoir.

[89] Site officiel, https://www.wampserver.com.