Vérification et validation formelles pour l'Internet des objets

Moez Krichen

Vérification et validation formelles pour l'Internet des objets

Moez Krichen

Laboratoire ReDCAD, Université de Sfax, Tunisie
moez.krichen@redcad.org

Résumé. L'Internet des objets (IdO) a ouvert une nouvelle ère de dispositifs et de systèmes connectés, avec des applications allant des soins de santé au transport. Cependant, la fiabilité et la sécurité de ces systèmes sont des préoccupations critiques qui doivent être abordées pour garantir leur fonctionnement sûr et efficace. Ce document présente une enquête approfondie sur les techniques de vérification et de validation formelles (V&V) pour les systèmes IdO, en mettant l'accent sur les défis et les problèmes ouverts dans ce domaine. Dans cette étude, nous explorons en détail les méthodes formelles et les techniques de test utilisées pour garantir la qualité et la robustesse des systèmes IdO. Nous examinons comment ces techniques peuvent être appliquées pour vérifier la conformité des dispositifs IdO aux spécifications et aux exigences fonctionnelles. Nous nous penchons également sur les méthodes de validation qui permettent de s'assurer que les systèmes IdO fonctionnent de manière fiable, sécurisée et efficace dans des scénarios réels. Un défi majeur dans la vérification des systèmes IdO est celui de l'explosion d'état, où le nombre de configurations possibles devient rapidement ingérable pour une vérification exhaustive. Nous analysons les différentes approches et techniques proposées pour surmonter ce défi, telles que l'abstraction, la modélisation formelle et l'utilisation de méthodes de test avancées. Par ailleurs, nous explorons comment l'intelligence artificielle (IA) peut être utilisée dans le domaine de la vérification et de la validation des systèmes IdO. Nous examinons les méthodes d'apprentissage automatique et d'apprentissage profond qui peuvent être appliquées pour améliorer l'efficacité et l'efficience des techniques de test et de vérification. Nous présentons des exemples d'outils et de frameworks qui intègrent l'IA dans le processus de V&V des systèmes IdO. Enfin, nous identifions et discutons des défis et des problèmes en suspens dans le domaine de la V&V pour l'IdO. Nous proposons des orientations futures possibles pour la recherche, notamment en ce qui concerne la modélisation formelle, les techniques de test avancées et l'intégration de l'IA dans les processus de V&V. L'objectif de cette enquête est de fournir une compréhension approfondie des techniques de V&V formelles pour les systèmes IdO, tout en soulignant les zones nécessitant des recherches et des développements supplémentaires. En consolidant les connaissances actuelles et en proposant des perspectives d'avenir,

cette étude vise à promouvoir des avancées significatives dans le domaine de la vérification et de la validation des systèmes IdO, contribuant ainsi à la création de systèmes IdO fiables, sécurisés et efficaces.

1 Introduction

L'Internet des objets (IdO) est devenu une partie indispensable de la vie moderne, connectant des milliards d'appareils à Internet et permettant une connectivité transparente, une automatisation et une surveillance en temps réel (79; 3; 40; 46). L'IdO a transformé diverses industries, notamment les soins de santé, les transports, les maisons intelligentes et l'automatisation industrielle, entraînant des améliorations significatives en termes d'efficacité, de productivité et de commodité (10; 13; 4; 93). Cependant, les faiblesses des systèmes IdO, telles que les ressources limitées, les appareils hétérogènes et le manque de normalisation, les rendent vulnérables aux pannes et aux attaques dangereuses (2; 85; 56; 102; 12; 50).

Ces vulnérabilités ont entraîné d'importantes pertes et dommages par le passé. Par exemple, en 2016, l'attaque du botnet Mirai a exploité les faiblesses de sécurité des appareils IdO pour lancer une attaque massive de déni de service distribué (DDoS), perturbant Internet dans le monde entier. L'attaque a causé une perte estimée à 323 000 dollars par heure pour les entreprises touchées, totalisant plus de 100 millions de dollars de dommages (16). Un autre exemple est le ver Stuxnet, qui visait les systèmes de contrôle industriel et a causé des dommages physiques aux centrifugeuses nucléaires en Iran. L'attaque aurait retardé le programme nucléaire iranien de deux ans et causé plus d'un milliard de dollars de dommages (19).

Pour garantir la fiabilité, la sécurité et la sûreté des systèmes IdO, il est crucial d'appliquer des techniques de vérification et de validation formelles (FV&V). Les méthodes formelles utilisent des techniques mathématiques pour modéliser et analyser les systèmes de manière rigoureuse (118; 36; 35; 47; 64; 48). Elles permettent aux concepteurs de systèmes de spécifier le comportement et les propriétés du système à l'aide de notations mathématiques précises, telles que des formules logiques et des machines d'état. Les méthodes formelles peuvent ensuite utiliser des outils automatisés pour analyser le comportement et les propriétés du système, tels que la vérification de la cohérence, de l'exhaustivité et de la correction (42). Elles peuvent également identifier les erreurs, les vulnérabilités et les attaques potentielles en explorant le comportement du système dans différents scénarios (111). Les méthodes formelles peuvent aider à détecter les erreurs de conception tôt dans le processus de développement et garantir que le système satisfait aux exigences et aux normes spécifiées.

Les tests sont une étape essentielle du processus de vérification et de validation, et diverses techniques ont été proposées pour tester efficacement les systèmes IdO (109; 51; 94). Par exemple, les techniques de test basées sur les modèles (MBT) génèrent automatiquement des cas de test à partir d'un modèle formel du système, ce qui peut aider à obtenir une meilleure couverture de test et réduire les efforts de test (33; 7; 73). Les techniques de test fuzz génèrent des données d'entrée aléatoires pour mettre le système à l'épreuve et identifier les vulnérabilités et les cas limites (100; 31). De plus, les techniques de test en boucle matériel-logiciel peuvent tester l'interaction entre le

logiciel et les composants matériels des systèmes IdO, ce qui peut aider à détecter les problèmes d'intégration et les problèmes de compatibilité (96; 119).

Le besoin de vérification par des méthodes formelles découle de l'importance de garantir la fiabilité et la sécurité des systèmes IdO, qui sont souvent déployés dans des domaines critiques tels que les soins de santé, les transports et le contrôle industriel. Les techniques de FV&V fournissent une approche systématique pour vérifier la correction de ces systèmes en modélisant mathématiquement leur comportement et en vérifiant rigoureusement qu'ils respectent leurs spécifications. Bien que les techniques de FV&V puissent prendre du temps, elles peuvent également réduire le coût global de développement en identifiant et en éliminant les problèmes potentiels dès le processus de conception. De plus, l'utilisation d'outils et de techniques automatisés peut aider à réduire l'effort manuel nécessaire pour la FV&V, la rendant ainsi plus applicable pour respecter les contraintes de délai de mise sur le marché.

L'application de mises à jour et de correctifs aux systèmes IdO peut potentiellement introduire de nouvelles vulnérabilités ou modifier le comportement du système. Pour remédier à cela, les techniques de FV&V peuvent être utilisées pour vérifier la correction des mises à jour et des correctifs avant qu'ils ne soient appliqués au système. Cela peut contribuer à garantir que le système reste fiable et sécurisé même après l'application des mises à jour et des correctifs. De plus, les techniques de FV&V peuvent être utilisées pour vérifier la correction des mises à jour et des correctifs eux-mêmes, réduisant ainsi le risque d'introduction de nouvelles vulnérabilités ou erreurs.

Le développement d'une méthodologie générique pour valider tous les types de dispositifs IdO en utilisant des techniques de FV&V est un domaine de recherche actif. Bien que les techniques de FV&V soient applicables à une large gamme de systèmes IdO, les techniques et les outils spécifiques utilisés peuvent varier en fonction des caractéristiques du système à vérifier. Les chercheurs ont développé des méthodologies pour des types spécifiques de dispositifs IdO, tels que les capteurs, les actionneurs et les protocoles de réseau. Cependant, développer une méthodologie générique qui peut être appliquée à tous les types de dispositifs IdO reste un défi.

La motivation d'utiliser les techniques de FV&V dans les systèmes IdO réside dans l'importance de garantir leur fiabilité et leur sécurité. Bien que d'autres techniques telles que les tests et la simulation puissent également être utilisées pour vérifier la correction des systèmes IdO, elles peuvent ne pas être suffisantes pour garantir leur fiabilité et leur sécurité dans tous les cas. Les techniques de FV&V fournissent une approche formelle et rigoureuse pour vérifier la correction des systèmes IdO et peuvent identifier les problèmes potentiels qui ne peuvent pas être détectés par d'autres moyens. De plus, les techniques de FV&V peuvent être utilisées pour vérifier la correction du système tout au long de son cycle de développement, de la conception au déploiement en passant par les mises à jour et les correctifs.

L'objectif de cet article est de fournir une étude complète des techniques de vérification formelle (FV), de validation et de test pour les systèmes IdO. Nous passerons en revue les méthodes et les outils de pointe pour la modélisation, la spécification, la vérification et le test des systèmes IdO. Nous discuterons également de leurs forces et de leurs limites, et identifierons les défis ouverts et les orientations de recherche futures dans ce domaine. En offrant une vision globale du paysage de la vérification formelle,

de la validation et du test pour les systèmes IdO, cet article vise à aider les chercheurs et les praticiens à développer des systèmes IdO plus sûrs, fiables et dignes de confiance.

Les principales contributions de cet article sont résumées ci-dessous :

— Présentation des techniques de FV&V pour les systèmes IdO.
— Discussion des défis et des problèmes ouverts dans la FV&V pour les systèmes IdO.
— Examen des méthodes formelles et des techniques de test pour les systèmes IdO.
— Exploration de l'utilisation de l'IA dans le test logiciel pour les systèmes IdO.
— Identification des domaines de recherche et de développement futurs dans la FV&V pour les systèmes IdO.

Le reste de l'article est structuré comme suit. La section 2 présente un aperçu des travaux connexes. Dans la section 3, nous discutons des notions préliminaires liées à l'IdO, y compris la définition de l'IdO, ses caractéristiques et les défis qu'il présente. La section 4 donne un aperçu des méthodes formelles pour l'IdO. Dans la section 5, nous explorons les techniques de test pour l'IdO, y compris l'importance du test, les différents types de test et les défis du test dans l'IdO. La section 6 examine l'utilisation de l'IA dans le test logiciel pour les systèmes IdO, y compris ses avantages et des exemples d'outils. Dans la section 7, nous discutons des défis et des problèmes ouverts dans la FV&V pour les systèmes IdO et présentons les orientations futures possibles pour la recherche. Enfin, nous concluons l'article dans la section 8, en résumant les principales contributions et leur importance, et en mettant en évidence les domaines de travail futurs.

2 Travaux connexes

Dans cette section, nous passons en revue plusieurs approches de vérification formelle proposées pour les protocoles IdO ces dernières années, notamment IdO Conflict Checker, la gestion des services basée sur la confiance, BiAgents* et les modèles formels des protocoles de messagerie populaires (66; 74; 61; 104; 68; 49). Nous discutons également des approches formelles pour définir la sémantique de modélisation des applications IdO et garantir la sécurité de la couche physique. Enfin, nous examinons VerificationTalk, un mécanisme de vérification des configurations des appareils et des réseaux pour éviter un déploiement incorrect des applications IdO.

L'article (43) a passé en revue les approches de vérification formelle pour de nombreux protocoles IdO. La vérification formelle est essentielle pour détecter les vulnérabilités dès le début, selon le rapport. Les auteurs détaillent les propriétés et les approches. Une étude approfondie de la littérature identifie quatre domaines d'application : (1) vérifications fonctionnelles, (2) vérifications de propriétés de sécurité, (3) idées pour de meilleurs schémas comprenant des vérifications de propriétés de sécurité a priori, et (4) vérifications de l'implémentation des protocoles. L'article couvre les propriétés de sécurité et les outils de protocole. Les auteurs décrivent également les vérifications de modèles typiques et discutent des difficultés et des limitations liées à l'IdO (71; 57; 72; 69; 83).

La contribution fondamentale de (**?**) est l'invention d'une approche formelle, IdO Conflict Checker (IdOC2), pour assurer la sécurité du comportement des contrôleurs

et des actionneurs dans l'Internet des objets face aux conflits. Le travail spécifie et met en œuvre des politiques de sécurité pour les contrôleurs, les actions et les événements déclencheurs en Prolog pour démontrer la complétude logique et la cohérence (59; 77; 87; 82; 90). L'environnement Matlab Simulink avec ses blocs de vérification de modèle intégrés est utilisé pour implémenter les politiques de détection (84; 76; 86; 89). Les auteurs ont utilisé Simulink pour développer un environnement domotique intelligent et ont démontré comment les conflits affectent les actions et les fonctionnalités correspondantes (78; 80). L'évolutivité, l'efficacité et la précision de la méthode sont évaluées dans un environnement simulé, démontrant son utilité potentielle dans les applications IdO du monde réel.

Le travail (9) représente et vérifie les composants de gestion de services basés sur la confiance dans l'Internet des objets à l'aide d'une méthode formelle basée sur la logique d'ordre supérieur (HOL). L'étude examine les difficultés de la composition des services IdO et les alternatives basées sur la confiance, qui surpassent celles qui ne reposent pas sur la confiance. Les auteurs proposent un système de confiance et de réputation pour identifier les fournisseurs de services appropriés (SP) pour les plans de composition de services et utilisent HOL pour valider les comportements variés dans le système de confiance et les procédures de calcul de la valeur de confiance. Les nœuds malveillants peuvent fausser le calcul de la valeur de confiance, entraînant une sélection incorrecte du SP lors de la composition des services.

L'article (110) propose un modèle BiAgents* (Agents Bigraphiques) pour formaliser la structure et le comportement des systèmes de l'Internet des objets. Selon l'article, bien qu'il y ait eu beaucoup de recherches sur les réseaux et les appareils IdO, la formalisation et l'évaluation des systèmes IdO en sont encore à leurs débuts. La spécification BiAgents* est encodée dans le langage Maude pour permettre aux systèmes IdO de se comporter de manière autonome. Un système intelligent d'évitement de collision illustre et évalue l'approche proposée. Il formalise et normalise les interactions complexes entre les appareils intelligents dans l'écosystème IdO, ce qui rend l'article important.

Le principal résultat du travail (18) est l'introduction d'un modèle formel basé sur l'algèbre des processus de passage de messages temporels pour le protocole MQTT (MQ Telemetry Transport) version 3.1. Cette étude prend des décisions de modélisation et met en évidence les incohérences dans la spécification originale du protocole. Après avoir effectué une analyse statique sur le modèle de protocole formel, les auteurs constatent que le protocole fournit une sémantique de livraison au plus une fois et au moins une fois aux abonnés, comme défini, pour les deux premiers modes de fonctionnement QoS. Cependant, les auteurs concluent que la sémantique du troisième mode de fonctionnement, le plus élevé, est inexacte à certains égards et au mieux peu claire dans d'autres. Enfin, les auteurs proposent d'améliorer la QoS du protocole à ce stade. Ce travail contribue de manière significative car il crée un modèle formel pour un protocole de messagerie populaire et met en évidence les moyens d'améliorer sa spécification.

L'article (30) propose une approche formelle basée sur le MDE (Model-Driven Engineering) pour définir la sémantique formelle de ThingML, un langage de modélisation pour les applications de l'Internet des objets. La variabilité des composants IdO et des protocoles de communication rend leur conception et leur développement difficiles. ThingML, un profil UML prometteur pour faire face à ces difficultés, manque de sé-

mantique stricte, ce qui le rend inapproprié pour la vérification formelle et l'analyse des architectures système (39). La logique de réécriture et Maude sont utilisés pour construire la sémantique formelle de ThingML, en implémentant tous les concepts et comportements. Un programme développé par les auteurs convertit automatiquement les spécifications ThingML en Maude, permettant l'utilisation de méthodes analytiques avancées telles que la simulation et les tests de modèles. Ce travail propose une correspondance sémantique entre les idées de ThingML et les constructions de Maude, définit et met en œuvre une sémantique opérationnelle pour le langage d'action de ThingML dans Maude, et fournit une étude de cas.

Le principal apport de l'étude (114) est un modèle formel basé sur la preuve Event-B de la sécurité de la couche physique de l'Internet des objets et des menaces, de l'analyse des exigences au niveau des objectifs. L'article souligne la nécessité de la sécurité dans les dispositifs IdO, en particulier ceux qui génèrent, collectent ou traitent des données sensibles, et affirme que prévenir les vulnérabilités est préférable à les identifier. Les auteurs proposent une approche formelle en trois étapes : la construction de la couche physique de l'IdO, la recherche de faiblesses en matière de sécurité et la détection d'attaques au niveau de la couche physique telles que le brouillage et le détournement d'adresse MAC. Pour démontrer la généralisabilité, un système IdO d'électrocardiogramme (ECG) et un système d'alarme incendie sont utilisés comme études de cas. Les responsabilités de preuve de l'outil de vérification de modèles Rodin et l'animateur ProB valident l'approche des auteurs. L'approche formelle de la sécurité de la couche physique de l'IdO et de la détection des attaques de ce travail peut contribuer à prévenir les violations de sécurité et à sécuriser les données sensibles.

Le cadre proposé dans (29) utilise la technique formelle Event-B pour développer une architecture sécurisée de l'Internet des objets et traiter les problèmes de sécurité et de confidentialité liés à l'IdO. Le rapport note que, bien que diverses conceptions novatrices d'IdO aient été présentées, elles sont encore confrontées à des problèmes de sécurité et de confidentialité, et la vérification formelle peut aider à identifier les failles dès le début. Les auteurs utilisent des propriétés d'Event-B telles que la vérification formelle, les vérifications fonctionnelles et les vérificateurs de modèles pour concevoir des attaques de spoofing formelles pour l'environnement IdO et obtenir une précision de l'architecture IdO en exécutant des simulations, des obligations de preuve et des vérifications d'invariants. La vérification formelle, les vérifications fonctionnelles et les vérificateurs de modèles ont été utilisés pour valider les modèles d'architecture IdO-EAA, qui ont automatiquement rempli 82,35% des responsabilités de preuve à l'aide des prouveurs Event-B. Enfin, l'étude recommande une infrastructure de sécurité IdO bien définie pour réduire les problèmes de sécurité de l'IdO. Cette étude est importante car elle offre une manière formelle de développer une architecture IdO sécurisée et de résoudre les problèmes de sécurité et de confidentialité de l'IdO.

L'article (52) a passé en revue de nombreux défis de sécurité de l'Internet des objets et a proposé la vérification formelle comme une approche prometteuse pour détecter les failles et garantir la sécurité. En raison des limitations de batterie et de traitement, les dispositifs IdO rendent difficile la détection en ligne des cyberattaques. Les auteurs suggèrent de pré-déployer le dispositif avec des tests de sécurité plus stricts pour réduire la surface d'attaque. L'étude aborde les défis de sécurité de l'IdO tels que

la validité fonctionnelle, les erreurs de code, l'analyse des canaux secondaires et les chevaux de Troie matériels. Les procédures de pointe utilisent des outils de vérification formelle pour détecter les vulnérabilités avant le déploiement du dispositif. Cet article est important car il passe en revue les vulnérabilités de sécurité des dispositifs IdO et présente la vérification formelle comme solution.

Le mécanisme de vérification VerificationTalk, qui vérifie les configurations des appareils et des réseaux pour prévenir le déploiement incorrect des applications de l'Internet des objets, est la principale contribution de (108). L'étude montre que dans un contexte à deux domaines, les développeurs d'applications peuvent mettre en œuvre des fonctions réseau inappropriées ou connecter des appareils IdO qui ne devraient jamais être liés, ce qui entraîne des activités de fonctions réseau incorrectes. BigraphTalk vérifie la configuration des appareils IdO et AFLtalk évalue les fonctions réseau. Les auteurs suggèrent une détection d'anomalies en ligne utilisant un moniteur d'exécution et hors ligne en utilisant American Fuzzy Lop (AFL). Le moniteur d'exécution peut intercepter les messages potentiellement dangereux ciblant les appareils IdO, et VerificationTalk offre des retours d'informations pour le débogage des problèmes. En trouvant des vulnérabilités de sécurité des applications réseau, VerificationTalk aide à concevoir des applications IdO sécurisées. Les auteurs montrent qu'en développant correctement le moteur d'exécution IdOtalk, la capacité de test d'AFLtalk est trois fois supérieure à celle des méthodes AFL typiques. Ce travail présente une technique pour éviter le déploiement inapproprié des applications IdO en validant les configurations dans les domaines de l'appareil et du réseau, ce qui permet le développement d'applications IdO sécurisées et fiables.

Ces approches de vérification formelle ont le potentiel de résoudre plusieurs défis auxquels sont confrontées les applications IdO et d'aider au développement de systèmes IdO sécurisés et fiables.

3 Préliminaires liés à l'IdO

L'Internet des objets a connu une évolution significative au cours des dernières décennies, passant de simples communications machine à machine (M2M) à un vaste réseau d'appareils et de systèmes interconnectés (5; 25; 103; 8; 99; 53). L'évolution de l'IdO a été stimulée par les avancées dans les technologies de communication, telles que les réseaux sans fil, les capteurs et l'informatique en nuage, ainsi que par la demande croissante d'automatisation et de surveillance en temps réel dans diverses industries (6; 92; 67; 55; 38).

L'une des principales caractéristiques de l'IdO est l'hétérogénéité, où des appareils de différents fabricants, avec des capacités et des ressources variables, doivent fonctionner ensemble de manière transparente. Cette hétérogénéité s'applique également aux données générées par ces appareils, qui peuvent avoir différents formats, structures et sémantiques. Par conséquent, les systèmes IdO doivent utiliser des protocoles et des formats normalisés pour garantir l'interopérabilité et la compatibilité entre les appareils et les réseaux. Une autre caractéristique de l'IdO est sa capacité de mise à l'échelle, où les systèmes IdO doivent être capables de gérer un grand nombre d'appareils et de données sans compromettre les performances et la fiabilité. Cette évolutivité

peut être réalisée grâce à des architectures distribuées qui peuvent évoluer de manière horizontale ou verticale, en fonction des besoins de l'application. Cependant, la mise à l'échelle des systèmes IdO peut également accroître leur complexité et introduire de nouveaux défis liés à la gestion des données, à la sécurité et à la confidentialité. Les systèmes IdO doivent également être résilients aux pannes et aux attaques, en raison de la nature critique de nombreuses applications, telles que les soins de santé et l'automatisation industrielle. Cette résilience peut être obtenue grâce à la redondance, à la tolérance aux pannes et aux mécanismes de reprise après sinistre qui peuvent garantir la disponibilité et l'intégrité des données et des services fournis par les systèmes IdO. Cependant, garantir la résilience peut également augmenter les coûts et la complexité des systèmes IdO, nécessitant des compétences et des outils spécialisés. Enfin, les systèmes IdO doivent être capables de fonctionner dans des environnements dynamiques et imprévisibles, tels que les environnements extérieurs ou les véhicules en mouvement. Cela nécessite que les systèmes IdO puissent s'adapter aux conditions changeantes, telles que les changements de la topologie du réseau, les interférences ou la mobilité. De plus, les systèmes IdO doivent être en mesure de traiter les données en temps réel, ce qui nécessite des algorithmes efficaces de traitement et d'analyse des données.

L'architecture de l'IdO se compose généralement de quatre couches : la couche de perception, la couche réseau, la couche intergiciel et la couche d'application. La couche de perception comprend les capteurs et les actionneurs qui collectent et manipulent les données du monde physique. Ces capteurs peuvent être de différents types, tels que des capteurs de température, de pression, de lumière et de mouvement, et peuvent être connectés au réseau à l'aide de différents protocoles de communication, tels que Zigbee, Wi-Fi et Bluetooth. La couche réseau comprend les protocoles de communication et les réseaux qui permettent aux appareils de se connecter et de communiquer entre eux. Cette couche peut inclure différents types de réseaux, tels que les réseaux cellulaires, les réseaux satellitaires et les réseaux ad hoc, en fonction des besoins de l'application. La couche intergiciel fournit les logiciels et les services nécessaires pour gérer les appareils et les données, tels que le stockage des données, le traitement et la sécurité. Cette couche peut inclure divers composants, tels que des courtiers de données, des files d'attente de messages et des outils d'analyse des données, qui peuvent faciliter l'intégration et l'analyse de données provenant de différentes sources. Enfin, la couche d'application comprend les logiciels et les services qui utilisent les données et les appareils pour fournir des services à valeur ajoutée, tels que les maisons intelligentes, la surveillance médicale et l'automatisation industrielle. Cette couche peut inclure différents types d'applications, tels que des applications web, des applications mobiles et des applications intégrées, en fonction de la plateforme cible et des exigences de l'utilisateur.

Malgré les nombreux avantages de l'IdO, il existe également plusieurs limitations qui doivent être prises en compte. L'une des principales limitations est le manque de normalisation, qui peut entraver l'interopérabilité et la compatibilité des différents systèmes IdO. L'absence d'un langage et d'une notation communs pour décrire les systèmes IdO peut également entraver l'adoption et l'intégration des méthodes formelles dans le processus de développement. Par conséquent, il est nécessaire de mener des efforts de normalisation pour établir un cadre commun de modélisation, de spécifica-

tion, de vérification et de test des systèmes IdO à l'aide de méthodes formelles. Une autre limitation est liée aux risques de sécurité et de confidentialité associés à l'IdO, en raison de la grande surface d'attaque et des vulnérabilités de certains appareils et réseaux. Les systèmes IdO peuvent être exposés à différents types d'attaques, telles que les attaques par déni de service, les violations de données et les attaques de logiciels malveillants, qui peuvent compromettre la confidentialité, l'intégrité et la disponibilité des données et des services fournis par les systèmes IdO. De plus, les systèmes IdO peuvent collecter des données sensibles, telles que des informations personnelles sur la santé, des informations financières et des données de localisation, qui peuvent être détournées si elles ne sont pas correctement protégées.

Les systèmes IdO sont également confrontés à des défis liés à la consommation d'énergie, car de nombreux appareils sont alimentés par batterie et doivent fonctionner pendant de longues périodes sans recharge. Cela nécessite que les systèmes IdO utilisent des techniques efficaces de gestion de l'énergie, telles que le duty cycling, les modes veille et la récupération d'énergie, qui peuvent prolonger la durée de vie de la batterie des appareils et réduire leur impact environnemental. Enfin, la complexité des systèmes IdO peut rendre leur développement, leur test et leur maintenance difficiles, nécessitant des compétences et des outils spécialisés. Les systèmes IdO peuvent impliquer plusieurs couches, appareils, protocoles et normes, ce qui peut augmenter les efforts de test et rendre difficile l'obtention d'une couverture de test complète. De plus, les systèmes IdO peuvent être soumis à des changements de besoins, de technologies et de réglementations, ce qui peut nécessiter des mises à jour et une maintenance fréquentes.

4 Méthodes formelles

Dans cette section, nous présenterons un aperçu concis des formes les plus répandues de techniques formelles actuellement disponibles pour la communauté de recherche (118; 36).

— **Interprétation abstraite (26) :** L'interprétation abstraite est une méthode formelle utilisée pour analyser et vérifier le comportement des programmes informatiques. Il s'agit d'une technique qui consiste à approximer le comportement d'un programme en abstrayant certains de ses détails. L'objectif de l'interprétation abstraite est de prouver qu'un programme satisfait certaines propriétés, telles que la sécurité, la vivacité ou la terminaison. L'interprétation abstraite fonctionne en définissant un ensemble de valeurs abstraites qui représentent les états possibles d'un programme. Ces valeurs abstraites sont définies de manière à surestimer l'ensemble des valeurs concrètes possibles. Cela permet à l'interprétation abstraite de raisonner sur le comportement d'un programme sans l'exécuter réellement. Un des principaux avantages de l'interprétation abstraite est qu'elle peut être utilisée pour analyser des programmes trop complexes pour être analysés avec d'autres méthodes. Cela est possible car l'interprétation abstraite peut raisonner sur le comportement d'un programme à un niveau d'abstraction supérieur, ce qui permet de traiter un espace d'états beaucoup plus grand.

— **Analyse statique sémantique (37) :** L'analyse statique sémantique est une méthode formelle utilisée pour analyser le comportement des programmes informatiques en examinant leur code source. L'objectif de l'analyse statique sémantique est de détecter les erreurs et les problèmes potentiels dans un programme avant son exécution. L'analyse statique sémantique fonctionne en analysant la syntaxe et la structure d'un programme pour en déduire sa signification. Cela se fait en construisant un modèle mathématique du comportement du programme, qui peut ensuite être utilisé pour raisonner sur ses propriétés. Un des avantages de l'analyse statique sémantique est qu'elle peut être appliquée tôt dans le processus de développement, ce qui permet de gagner du temps et des ressources. En détectant les erreurs avant l'exécution d'un programme, l'analyse statique sémantique peut contribuer à garantir que le produit final est correct et fiable. Cependant, l'analyse statique sémantique peut être difficile car elle repose sur la capacité à raisonner sur des modèles mathématiques complexes de comportement de programme. Cela nécessite des connaissances spécialisées et une expertise en méthodes formelles et en analyse mathématique. De plus, la précision de l'analyse statique sémantique dépend de la qualité du modèle utilisé, ce qui peut être difficile à construire pour des programmes complexes (106; 1).

— **Vérification de modèles (101) :** La vérification de modèles est une méthode formelle utilisée pour vérifier la correction d'un système en explorant exhaustivement ses comportements possibles. Elle fonctionne en construisant un modèle du système et en spécifiant les propriétés souhaitées que le système doit satisfaire. Le vérificateur de modèles explore alors systématiquement tous les états possibles du système pour déterminer si ces propriétés sont satisfaites pour tous les comportements possibles. La vérification de modèles est particulièrement utile pour vérifier des systèmes complexes, tels que des systèmes concurrents et distribués, où les méthodes traditionnelles de test peuvent ne pas suffire. Elle peut également être utilisée pour vérifier des conceptions matérielles et des protocoles. L'un des principaux avantages de la vérification de modèles est qu'elle peut fournir une couverture complète de tous les comportements possibles, ce qui en fait un outil puissant pour garantir la correction de systèmes critiques.

— **Assistance à la preuve (34) :** Les assistants à la preuve sont des outils logiciels qui aident les utilisateurs à construire et à vérifier des preuves mathématiques. Ils fournissent un langage formel pour exprimer des énoncés mathématiques et un ensemble de règles pour manipuler ces énoncés afin de construire des preuves. Les assistants à la preuve sont utiles pour formaliser des théories mathématiques et vérifier leur correction. Ils peuvent également être utilisés pour vérifier la correction de logiciels et de conceptions matérielles. L'un des principaux avantages des assistants à la preuve est qu'ils fournissent un niveau élevé d'assurance que la preuve est correcte, car la preuve est construite à l'aide de règles formelles et le logiciel vérifie la correction de la preuve.

— **Vérification déductive (105) :** La vérification déductive est une méthode formelle utilisée pour vérifier la correction d'un logiciel en construisant une preuve formelle que le logiciel satisfait ses spécifications. Elle fonctionne en commen-

çant par les spécifications du logiciel, puis en construisant systématiquement une preuve que la mise en œuvre du logiciel satisfait ces spécifications. La vérification déductive est particulièrement utile pour garantir la correction des systèmes critiques en matière de sécurité, tels que ceux utilisés dans l'aviation et les dispositifs médicaux. L'un des principaux avantages de la vérification déductive est qu'elle peut fournir un niveau élevé d'assurance que le logiciel est correct, car la preuve est construite à l'aide de règles formelles et la preuve peut être vérifiée par un ordinateur.

— **Conception par raffinement (23; 21) :** La conception par raffinement est une méthode formelle utilisée pour développer des logiciels corrects en affinant de manière itérative une spécification abstraite du logiciel jusqu'à obtenir une implémentation détaillée. Elle fonctionne en commençant par une spécification de haut niveau du logiciel, puis en affinant cette spécification étape par étape jusqu'à obtenir une implémentation détaillée. La conception par raffinement peut aider à garantir que le logiciel satisfait ses spécifications et est exempt d'erreurs. Elle peut également aider à garantir que le logiciel est maintenable et peut être facilement modifié lorsque les exigences changent. L'un des principaux avantages de la conception par raffinement est qu'elle offre une approche systématique du développement de logiciels qui peut contribuer à garantir que le produit final est correct et satisfait ses spécifications.

— **Tests basés sur les modèles (MBT) (60; 58; 88; 70)** : Le MBT est une méthode formelle utilisée pour tester des logiciels en générant des cas de test à partir d'un modèle du logiciel. Elle fonctionne en construisant un modèle du logiciel, puis en utilisant ce modèle pour générer automatiquement des cas de test qui explorent différentes parties du logiciel. Le MBT peut aider à garantir que le logiciel satisfait ses spécifications et est exempt d'erreurs. Il peut également contribuer à réduire le temps et les efforts nécessaires pour tester le logiciel. L'un des principaux avantages du MBT est qu'il offre une approche systématique des tests qui peut contribuer à garantir que le produit final est correct et satisfait ses spécifications.

De plus, les méthodes formelles peuvent être classées en trois catégories : Complètes, Partielles et Asymptotiquement Complètes. Dans cette classification, les méthodes complètes sont garanties de fournir une réponse définitive à un problème donné, tandis que les méthodes partielles peuvent ne pas fournir une réponse définitive mais peuvent fournir des informations utiles. Les méthodes asymptotiquement complètes ne sont pas garanties de trouver une solution, mais lorsque la taille du problème augmente, la probabilité de trouver une solution approche 1. Voici quelques détails supplémentaires concernant ces trois classes :

— **Méthodes Complètes (27)** : Les méthodes complètes sont des méthodes formelles garanties de fournir une réponse définitive à un problème donné. Cela signifie que si un problème a une solution, une méthode complète la trouvera. Par exemple, la vérification de modèles est une méthode complète car elle peut explorer systématiquement tous les comportements possibles d'un système pour déterminer si une propriété donnée est vérifiée ou non. Différents types de mé-

thodes complètes existent, à savoir : les méthodes basées sur la SMT [1] (20), les méthodes basées sur le MILP [2] (117), les méthodes basées sur l'optimisation, etc.

— **Méthodes Partielles (28)** : Les méthodes partielles sont des méthodes formelles qui peuvent ne pas fournir une réponse définitive à un problème donné. Cela signifie qu'une méthode partielle peut ne pas être en mesure de déterminer si un problème a une solution ou non. Par exemple, l'interprétation abstraite est une méthode partielle car elle peut fournir une surestimation du comportement d'un programme, mais elle peut ne pas être en mesure de déterminer si le programme satisfait une propriété donnée ou non.

— **Méthodes Asymptotiquement Complètes** : Les méthodes asymptotiquement complètes sont des méthodes formelles qui ne garantissent pas de fournir une réponse définitive à un problème, mais lorsque la taille du problème augmente, la probabilité de trouver une solution approche 1. Cela signifie que pour des problèmes très grands, une méthode asymptotiquement complète trouvera presque toujours une solution. Par exemple, la recherche heuristique est une méthode asymptotiquement complète car lorsque la taille de l'espace de recherche augmente, la probabilité de trouver une solution approche 1, même s'il n'y a aucune garantie de trouver une solution pour une instance de problème donnée.

Les méthodes complètes fournissent une réponse définitive mais peuvent être computationnellement coûteuses et ne se prêtent pas toujours bien aux problèmes de grande taille. Les méthodes partielles peuvent gérer des problèmes complexes mais peuvent ne pas être en mesure de détecter toutes les erreurs ou de trouver toutes les solutions. Les méthodes asymptotiquement complètes peuvent gérer des problèmes très grands et trouver rapidement des solutions, mais elles ne garantissent pas toujours de trouver une solution et ne peuvent pas garantir la correction. Comprendre les points forts et les limites de ces différents types de méthodes formelles peut aider les développeurs à choisir la méthode la plus appropriée pour leur problème spécifique.

Les avantages les plus importants de l'utilisation des approches formelles sont les suivants :

— **Abstraction :** Les approches formelles permettent l'abstraction, ce qui signifie qu'elles peuvent fournir une vision plus globale du système logiciel. Cela peut aider à gérer la complexité en masquant les détails non pertinents et en se concentrant sur les caractéristiques essentielles du système. L'abstraction facilite également la réflexion sur le comportement du système et l'identification d'éventuelles erreurs et défauts.

— **Analyse rigoureuse :** Les méthodes formelles fournissent une approche rigoureuse et systématique pour analyser les systèmes logiciels. Cela signifie qu'elles utilisent des modèles mathématiques et des techniques bien définis pour analyser le logiciel, ce qui peut contribuer à garantir que l'analyse est précise et complète. L'analyse rigoureuse peut identifier des défauts et des erreurs qui pourraient être négligés par d'autres méthodes, telles que les tests ou les revues informelles.

1. SMT = Satisfiability Modulo Theory.
2. MILP = Mixed Integer Linear Programming.

— **Détection précoce des défauts :** Les méthodes formelles peuvent être appliquées tôt dans le processus de développement logiciel, ce qui peut aider à identifier les défauts et les erreurs avant qu'ils ne deviennent plus difficiles et coûteux à corriger. La détection précoce des défauts peut également contribuer à améliorer la qualité globale du logiciel et à réduire le risque de défauts pouvant entraîner des défaillances du système ou des risques pour la sécurité.
— **Garanties de correction :** Les méthodes formelles peuvent fournir des garanties de correction, ce qui signifie qu'elles peuvent prouver que le logiciel respecte ses spécifications et se comporte correctement. Cela peut offrir un niveau élevé d'assurance quant à la correction et à la fiabilité du logiciel. Les garanties de correction sont particulièrement importantes pour les systèmes critiques en termes de sécurité, où des erreurs ou des défauts pourraient avoir des conséquences graves.
— **Fiabilité :** Les méthodes formelles peuvent améliorer la fiabilité des systèmes logiciels en réduisant le risque d'erreurs et de défauts. Cela peut contribuer à garantir que le logiciel se comporte comme prévu et qu'il est robuste et résistant aux entrées ou aux conditions inattendues. La fiabilité est particulièrement importante pour les systèmes qui doivent fonctionner en continu ou qui ne peuvent pas être facilement réparés ou remplacés.
— **Scénarios de test efficaces :** Les méthodes formelles peuvent aider à identifier les scénarios de test les plus efficaces pour un système logiciel. Cela peut réduire le temps et les efforts nécessaires pour tester le logiciel, tout en garantissant que le logiciel respecte ses spécifications et se comporte correctement. Les scénarios de test efficaces peuvent également contribuer à améliorer la qualité globale du logiciel et à réduire le risque de défauts pouvant entraîner des défaillances du système ou des risques pour la sécurité.
— **Maintenabilité :** Les méthodes formelles peuvent améliorer la maintenabilité des systèmes logiciels en fournissant une spécification claire et précise du comportement du système. Cela peut faciliter la modification ou la refonte du logiciel sans introduire d'erreurs ou de défauts. Les méthodes formelles peuvent également contribuer à garantir que les modifications ne violentpas les spécifications ou les exigences du système.
— **Réutilisabilité :** Les méthodes formelles peuvent améliorer la réutilisabilité des composants logiciels en fournissant une spécification claire et précise de leur comportement. Cela signifie que les composants logiciels peuvent être réutilisés dans différents contextes sans introduire d'erreurs ou de défauts. Les méthodes formelles peuvent également contribuer à garantir que les composants réutilisés se comportent correctement dans tous les contextes.
— **Standardisation :** Les méthodes formelles peuvent fournir une approche standardisée du développement et de la vérification des logiciels. Cela signifie que les systèmes logiciels peuvent être développés et vérifiés à l'aide d'un ensemble commun de techniques et d'outils, ce qui peut améliorer l'interopérabilité et réduire le risque d'erreurs ou de problèmes de compatibilité.
— **Confiance :** Les méthodes formelles peuvent donner aux développeurs et aux parties prenantes confiance dans la correction et la fiabilité du système logiciel.

Cela peut accroître la confiance dans le système logiciel et réduire le risque de conséquences négatives, telles que des défaillances du système ou des risques pour la sécurité.

5 Techniques de test

Le test (Testing) est une partie essentielle du processus de développement de logiciels, car il contribue à garantir que les systèmes logiciels sont fiables, performants et répondent aux besoins de leurs utilisateurs. Il existe de nombreuses approches différentes pour tester les logiciels, chacune ayant ses propres objectifs et procédures spécifiques :

- **Tests unitaires (54) :** Les tests unitaires sont une méthode de test qui se concentre sur les unités individuelles ou les composants d'un système logiciel. L'objectif des tests unitaires est de garantir que chaque composant du système fonctionne comme prévu et respecte ses normes. Les tests unitaires sont généralement réalisés en créant et en exécutant des cas de test pour chaque unité. Les tests unitaires ont l'avantage principal de détecter les erreurs et les défauts tôt dans le processus de développement, ce qui les rend plus faciles et moins coûteux à corriger. Le plus grand inconvénient des tests unitaires est qu'ils peuvent ne pas révéler les défauts ou les erreurs qui se produisent lorsque les unités sont fusionnées.
- **Tests d'intégration (107) :** Les tests d'intégration sont une méthode de test qui se concentre sur les interactions de plusieurs unités ou composants d'un système logiciel. L'objectif des tests d'intégration est de garantir que le système dans son ensemble fonctionne comme prévu et que les unités fonctionnent correctement lorsqu'elles sont combinées. Les tests d'intégration sont souvent réalisés en testant différentes combinaisons d'unités et en vérifiant qu'elles fonctionnent comme prévu. Le principal avantage des tests d'intégration est qu'ils peuvent révéler les défauts ou les erreurs qui se produisent lorsque les unités sont fusionnées, ce qui les rend plus faciles et moins coûteux à corriger. Le plus grand inconvénient des tests d'intégration est qu'ils peuvent ne pas détecter les défauts ou les problèmes qui se produisent lorsque le système est soumis à un stress ou à une charge élevée.
- **Tests d'acceptation (116) :** Les tests d'acceptation sont une méthode permettant de déterminer si un système logiciel répond à ses exigences et spécifications. L'objectif des tests d'acceptation est de garantir que le système est acceptable pour ses parties prenantes et qu'il répond à leurs exigences. Les tests d'acceptation sont souvent réalisés en soumettant le système à une utilisation réelle et en vérifiant s'il répond aux exigences et aux spécifications. Les tests d'acceptation ont l'avantage principal de garantir que le système répond aux besoins de ses parties prenantes. Le principal inconvénient des tests d'acceptation est qu'ils peuvent ne pas révéler les erreurs ou les problèmes qui se produisent lorsque le système est soumis à un stress ou à une charge élevée.
- **Tests fonctionnels (115) :** Les tests fonctionnels sont une approche de test qui se concentre sur la fonctionnalité d'un système logiciel. L'objectif des tests

fonctionnels est de garantir que le système fonctionne correctement et qu'il répond à ses exigences et spécifications. Les tests fonctionnels sont généralement réalisés en testant le système par rapport à un ensemble de cas de test prédéfinis qui couvrent tous les aspects de sa fonctionnalité. Le principal avantage des tests fonctionnels est qu'ils peuvent garantir que le système fonctionne correctement et qu'il répond à ses exigences et spécifications.

— **Tests de convivialité (41) :** Les tests de convivialité sont une approche de test qui se concentre sur la facilité d'utilisation d'un système logiciel. L'objectif des tests de convivialité est de garantir que le système est utilisable et qu'il répond aux besoins de ses utilisateurs. Les tests de convivialité sont généralement réalisés en testant le système avec un groupe d'utilisateurs représentatifs et en observant leur interaction avec le système. Le principal avantage des tests de convivialité est qu'ils peuvent garantir que le système est facile à utiliser et qu'il répond aux besoins de ses utilisateurs.

— **Tests de stress (91) :** Les tests de stress sont une approche de test qui se concentre sur la manière dont un système logiciel se comporte sous stress ou charge élevée. L'objectif des tests de stress est de garantir que le système peut gérer de grands volumes de trafic ou de demandes sans planter ou échouer. Les tests de stress sont généralement réalisés en soumettant le système à un volume élevé de trafic ou de demandes et en observant son comportement. Le principal avantage des tests de stress est qu'ils peuvent garantir que le système est fiable et peut gérer de grands volumes de trafic ou de demandes.

— **Tests de performance (11) :** Les tests de performance sont une méthode permettant de déterminer le bon fonctionnement d'un système logiciel dans des conditions d'utilisation normales. L'objectif des tests de performance est de garantir que le système est réactif et répond aux besoins de ses utilisateurs. Les tests de performance sont généralement réalisés en soumettant le système à une charge représentative et en évaluant son fonctionnement. Le principal avantage des tests de performance est qu'ils garantissent que le système est réactif et fonctionne correctement pour ses utilisateurs. Le principal inconvénient des tests de performance est qu'ils peuvent ne pas détecter les erreurs ou les problèmes qui se produisent lorsque le système est soumis à un stress ou à une charge élevée.

— **Tests de régression (81) :** Les tests de régression sont une méthode de test qui vise à déterminer si les modifications apportées à un système logiciel ont introduit de nouvelles erreurs ou défauts. L'objectif des tests de régression est de garantir que le système continue de fonctionner correctement après les modifications qui lui ont été apportées. Les tests de régression sont souvent réalisés en retestant le système par rapport à un ensemble de cas de test spécifiés après les modifications. Le principal avantage des tests de régression est qu'ils garantissent que le système continue de fonctionner correctement après les modifications. Le principal inconvénient des tests de régression est qu'ils peuvent ne pas révéler les erreurs ou les problèmes qui se produisent lorsque le système est soumis à un stress ou à une pression.

6 Utilisation de l'IA dans les tests de logiciels

L'utilisation de l'intelligence artificielle (IA) (44; 63; 15; 75; 97; 62; 95; 65; 98; 14; 22) dans les tests de logiciels gagne en popularité ces dernières années (32; 113). L'IA peut aider à automatiser diverses tâches liées aux tests de logiciels, telles que la génération de cas de test, l'exécution des tests et l'analyse des résultats. L'un des principaux avantages de l'utilisation de l'IA dans les tests de logiciels est la capacité à améliorer la couverture et la qualité des tests, car l'IA peut analyser de grandes quantités de données et identifier des motifs et des anomalies qui ne seraient peut-être pas apparents aux testeurs humains (32). Cela peut permettre une meilleure détection des défauts et des vulnérabilités, réduisant ainsi le risque de défaillances logicielles et de temps d'arrêt.

Un autre avantage de l'utilisation de l'IA dans les tests de logiciels est la capacité à réduire le temps et les efforts nécessaires pour les tests (113). L'IA peut automatiser les tâches répétitives et chronophages, telles que les tests de régression, permettant aux testeurs de se concentrer sur des tâches plus complexes et créatives, telles que les tests exploratoires. Cela peut entraîner des cycles de publication plus rapides et un temps de mise sur le marché réduit, ce qui est crucial dans l'industrie du développement de logiciels d'aujourd'hui, où tout va très vite.

De plus, l'IA peut également contribuer à améliorer l'efficacité et l'efficacité des équipes de test, car elle peut fournir des informations et des recommandations basées sur l'analyse des données et des algorithmes d'apprentissage automatique (32). Cela peut aider les testeurs à hiérarchiser leurs efforts de test et à se concentrer sur les domaines les plus critiques et les plus susceptibles de présenter des défauts. De plus, l'IA peut également contribuer à réduire les coûts des tests, car elle peut identifier les défauts et les vulnérabilités dès les premières phases du cycle de développement, réduisant ainsi la nécessité de travaux de correction et de maintenance coûteux.

6.1 Avantages

L'utilisation de l'IA dans les tests de logiciels offre de nombreux avantages qui peuvent contribuer à améliorer l'efficacité, l'efficacité et la qualité du développement de logiciels. Dans cette section, nous discuterons de certains des principaux avantages qui peuvent être obtenus grâce à l'utilisation de stratégies d'IA pour les tests de logiciels.

— **Écriture automatique de cas de test :** L'écriture automatique de cas de test est l'un des avantages les plus importants de l'utilisation de l'IA dans les tests de logiciels. L'IA peut analyser le code et identifier les zones potentielles de faiblesse, ce qui lui permet de générer des cas de test qui peuvent tester en profondeur le logiciel. Cela peut permettre de gagner un temps considérable et des efforts qui seraient sinon consacrés à l'écriture manuelle des cas de test. De plus, les cas de test générés par l'IA peuvent souvent couvrir plus de scénarios et de cas particuliers que les cas de test rédigés par des humains, ce qui permet d'effectuer des tests plus approfondis et d'obtenir une meilleure qualité de logiciel.

— **Mise sur le marché rapide :** L'utilisation de l'IA dans les tests de logiciels peut contribuer à réduire le délai de mise sur le marché des produits logiciels. En automatisant des tâches répétitives et chronophages telles que les tests de

régression, l'IA peut accélérer le processus de test. Cela peut aider les entreprises de logiciels à commercialiser leurs produits plus rapidement, ce qui leur confère un avantage concurrentiel. De plus, une mise sur le marché plus rapide peut entraîner une augmentation des revenus et une meilleure satisfaction des clients, car les clients ont tendance à choisir des produits qui sont lancés rapidement et régulièrement mis à jour avec de nouvelles fonctionnalités.

— **Réponse/retour d'information précoce :** Un autre avantage de l'utilisation de l'IA dans les tests de logiciels est la capacité à fournir des commentaires précoces sur la qualité du logiciel. L'IA peut détecter les défauts et les vulnérabilités dès les premières phases du cycle de développement, ce qui permet aux développeurs de les traiter avant qu'ils ne deviennent des problèmes majeurs. Cela peut contribuer à améliorer la qualité et la fiabilité du logiciel, ce qui se traduit par une meilleure satisfaction des clients. De plus, une détection précoce des problèmes peut contribuer à réduire les coûts et les efforts nécessaires pour les corriger ultérieurement dans le processus de développement.
— **Analyse prédictive :** L'analyse prédictive est un autre avantage de l'utilisation de l'IA dans les tests de logiciels. L'IA peut analyser des données historiques et en temps réel pour prédire le comportement futur d'un système logiciel. Cela peut aider à identifier les problèmes potentiels avant qu'ils ne se produisent, ce qui permet aux développeurs de prendre des mesures préventives pour éviter les temps d'arrêt ou les défaillances du système. De plus, l'analyse prédictive peut contribuer à optimiser les performances et l'efficacité des systèmes logiciels, ce qui se traduit par de meilleures expériences utilisateur et une meilleure satisfaction des clients.
— **Plateforme intégrée :** L'utilisation de l'IA dans les tests de logiciels peut contribuer à intégrer différents outils et plates-formes de test. L'IA peut contribuer à unifier différentes méthodes de test, telles que les tests unitaires, les tests d'intégration et les tests système. Cela peut aider les entreprises de logiciels à gagner du temps et à réduire les coûts en utilisant une plateforme de test unique et intégrée. De plus, une plateforme de test intégrée peut fournir une vue d'ensemble de la qualité du logiciel, ce qui permet aux développeurs d'identifier et de résoudre les problèmes de manière plus efficace.
— **Réduction des tests basés sur l'interface utilisateur :** L'IA peut aider à réduire la nécessité des tests basés sur l'interface utilisateur, qui sont souvent longs et coûteux. En automatisant les tests en arrière-plan, l'IA peut aider à identifier les problèmes sans avoir besoin de tests basés sur l'interface utilisateur, ce qui réduit l'effort global de test requis. De plus, la réduction des tests basés sur l'interface utilisateur peut améliorer l'efficacité des équipes de test, leur permettant de se concentrer sur des tâches de test plus complexes et critiques.
— **Meilleure couverture du code :** L'IA peut contribuer à améliorer la couverture du code en identifiant les parties du logiciel qui ne sont pas suffisamment couvertes par les cas de test existants. Cela peut contribuer à garantir que toutes les parties du logiciel sont rigoureusement testées, réduisant ainsi les risques de problèmes et de vulnérabilités. De plus, une meilleure couverture du code peut améliorer la qualité et la fiabilité du logiciel, ce qui améliore l'expérience utili-

sateur globale et la satisfaction des clients.

— **Amélioration de la fiabilité :** En automatisant les tâches de test, l'IA peut contribuer à améliorer la fiabilité des produits logiciels. Les tests automatisés peuvent détecter des défauts et des vulnérabilités qui pourraient être manqués lors de tests manuels, ce qui conduit à des produits logiciels plus fiables et stables. De plus, une amélioration de la fiabilité peut contribuer à réduire les coûts et les efforts nécessaires pour la maintenance et le support, ce qui améliore l'efficacité et l'efficacité globales des équipes de développement de logiciels.

— **Amélioration de la qualité :** L'utilisation de l'IA dans les tests de logiciels peut contribuer à améliorer la qualité globale des produits logiciels. En détectant les défauts et les vulnérabilités tôt dans le cycle de développement, l'IA peut contribuer à garantir que les produits logiciels sont de haute qualité et répondent aux attentes des clients. De plus, une amélioration de la qualité peut entraîner une meilleure satisfaction des clients, une augmentation des revenus et un avantage concurrentiel sur le marché.

— **Tests automatisés de validation visuelle :** L'IA peut également être utilisée pour des tests automatisés de validation visuelle, qui consistent à comparer la sortie visuelle d'un système logiciel avec les résultats attendus. Cela peut contribuer à identifier les défauts visuels et les incohérences, améliorant ainsi la qualité globale et l'expérience utilisateur du logiciel. De plus, les tests automatisés de validation visuelle peuvent contribuer à réduire l'effort requis pour les tests visuels manuels, permettant aux équipes de test de se concentrer sur des tâches de test plus complexes et critiques.

En résumé, l'utilisation de l'IA dans les tests de logiciels offre de nombreux avantages, notamment une mise sur le marché plus rapide, une meilleure fiabilité et qualité, ainsi qu'une réduction des efforts et des coûts de test. À mesure que la technologie de l'IA continue d'évoluer, il est probable que d'autres avantages deviendront apparents, ce qui en fera un outil de plus en plus précieux pour le développement et les tests de logiciels.

6.2 Exemples d'outils

Il existe de nombreux outils de test basés sur l'IA disponibles sur le marché qui peuvent aider à améliorer l'efficacité et l'efficacité des tests logiciels. Ces outils utilisent l'IA pour automatiser diverses tâches de test, telles que la génération de cas de test, l'exécution des tests et la détection des défauts. Voici quelques exemples :

— **Applitools** est un outil de test visuel qui utilise l'IA pour détecter automatiquement les défauts visuels et les incohérences dans les applications web et mobiles. En utilisant des algorithmes de vision par ordinateur, Applitools peut comparer des captures d'écran d'une application sur différents appareils, navigateurs et résolutions pour identifier les différences qui peuvent indiquer un défaut. L'outil peut s'intégrer à des frameworks de test populaires tels que Selenium et Appium pour incorporer facilement les tests visuels dans les processus existants. Applitools fournit également un tableau de bord qui met en évidence les problèmes visuels et facilite le suivi et la gestion des défauts. Les testeurs peuvent utiliser Applitools pour améliorer la couverture et la précision de leurs tests visuels,

ce qui conduit à de meilleurs produits logiciels et une satisfaction accrue des clients.

— **Appvance IQ** est un outil de test basé sur l'IA qui utilise des algorithmes d'apprentissage automatique pour générer et exécuter automatiquement des cas de test sur plusieurs plateformes et environnements. L'outil peut analyser le comportement des utilisateurs pour générer des cas de test couvrant les cas d'utilisation les plus critiques et les plus courants. Appvance IQ peut également détecter les défauts et les vulnérabilités et fournir des recommandations pour améliorer la qualité logicielle. L'outil fournit un tableau de bord qui facilite le suivi et la gestion des défauts, ainsi que des rapports détaillés et des analyses sur les activités de test. Les testeurs peuvent optimiser leur couverture et leur précision de test tout en économisant du temps et des efforts sur la création et la maintenance des cas de test en utilisant Appvance IQ.
— **Functionize** est un outil de test basé sur l'IA qui permet aux testeurs de générer et d'exécuter de manière autonome des cas de test et de détecter et de hiérarchiser les défauts. En utilisant des algorithmes avancés d'apprentissage automatique, Functionize peut analyser le comportement des utilisateurs pour générer des cas de test couvrant les cas d'utilisation critiques et courants et hiérarchiser automatiquement les défauts en fonction de leur gravité. Functionize fournit un tableau de bord qui facilite le suivi et la gestion des défauts, ainsi que des rapports détaillés et des analyses sur les activités de test.
— **Mabl** est un outil de test basé sur l'IA qui permet aux testeurs d'identifier et de hiérarchiser automatiquement les problèmes et de générer et de maintenir des cas de test. L'outil utilise des algorithmes avancés d'apprentissage automatique pour analyser le comportement des utilisateurs et générer des cas de test couvrant les cas d'utilisation critiques et courants. Mabl peut également détecter les problèmes et les vulnérabilités et les hiérarchiser en fonction de leur gravité, réduisant ainsi les efforts nécessaires pour le triage manuel des défauts. L'outil fournit un tableau de bord qui facilite le suivi et la gestion des défauts, ainsi que des rapports détaillés et des analyses sur les activités de test.
— **ReTest** est une solution de test basée sur l'intelligence artificielle qui permet aux testeurs d'évaluer les exigences logicielles et de produire des cas de test couvrant toutes les combinaisons potentielles de paramètres d'entrée. Le programme analyse les exigences et génère des cas de test couvrant toutes les combinaisons concevables de paramètres d'entrée, garantissant une couverture de test complète. ReTest peut également trouver automatiquement des défauts en exécutant les cas de test générés et fournir des informations détaillées sur les problèmes détectés. L'outil permet aux testeurs d'optimiser leur processus de test en générant automatiquement des cas de test complets et en identifiant rapidement les problèmes, ce qui permet de gagner du temps et des ressources.

Ces exemples d'outils de test basés sur l'IA ne sont qu'une petite sélection parmi de nombreuses autres solutions disponibles sur le marché. Chaque outil a ses propres fonctionnalités et avantages, il est donc important d'évaluer attentivement les besoins spécifiques de votre projet et de choisir l'outil qui convient le mieux à votre contexte.

7 Défis et problèmes ouverts

L'Internet des objets présente un ensemble unique de défis pour les techniques de vérification et de validation (V&V). L'un des principaux défis est la complexité et l'hétérogénéité des systèmes IdO. Les systèmes IdO peuvent impliquer de nombreux appareils et réseaux, chacun ayant un matériel, un logiciel et des protocoles de communication différents. Cela rend difficile le développement d'un cadre V&V unifié qui puisse être appliqué à tous les appareils. De plus, le manque de normalisation dans les appareils et les réseaux IdO rend difficile le développement de modèles formels qui capturent avec précision le comportement de ces systèmes. Par exemple, différents appareils peuvent avoir des protocoles de communication différents ou utiliser différents formats de données, ce qui rend difficile le développement d'un modèle formel unifié qui puisse être appliqué à tous les appareils. Un autre défi est la nature dynamique des systèmes IdO. Les appareils peuvent rejoindre ou quitter le système à tout moment, et le comportement du système peut changer en fonction du contexte et de l'environnement. Cela rend difficile le développement d'un modèle formel statique qui puisse capturer avec précision le comportement du système. Pour relever ces défis, les chercheurs ont développé des modèles formels spécifiques aux appareils et des techniques de vérification qui peuvent être adaptés aux caractéristiques spécifiques de chaque appareil. De plus, ils ont développé des interfaces et des protocoles standardisés qui permettent l'interopérabilité entre les appareils et les réseaux, améliorant ainsi la précision et la fiabilité des modèles formels.

Un autre défi dans les techniques de V&V pour l'IdO est le problème de l'explosion de l'état. Les systèmes IdO peuvent impliquer un grand nombre d'appareils et d'états, ce qui rend difficile leur analyse exhaustive. Ce problème peut être résolu en utilisant des techniques d'abstraction, de modularisation et de détection de symétrie. L'abstraction consiste à simplifier le modèle du système en supprimant les fonctionnalités superflues, tandis que la modularisation consiste à diviser la vérification de systèmes complexes en sous-problèmes plus petits. La détection de symétrie consiste à réduire l'espace d'états en identifiant les symétries qui se produisent pendant l'exécution du système et en créant une correspondance entre les états et les représentants des classes d'équivalence. Ces techniques ont été utilisées dans des recherches précédentes pour résoudre le problème de l'explosion de l'état dans la V&V des systèmes IdO. Cependant, il est encore nécessaire de développer des techniques plus efficaces et plus efficaces qui puissent gérer la nature dynamique et hétérogène des systèmes IdO.

Un autre défi dans la V&V des systèmes IdO est la nécessité de garantir qu'ils répondent aux exigences de performance tout en maintenant la sécurité et la fiabilité. De nombreux systèmes IdO sont utilisés dans des applications critiques en termes de sécurité, telles que les soins de santé et les transports, où la fiabilité et la sécurité revêtent une importance primordiale. De plus, les systèmes IdO impliquent souvent des contraintes temps réel, ce qui peut rendre difficile la garantie qu'ils répondent aux exigences de performance. Cela peut être résolu en utilisant des modèles formels tels que les automates temporisés et d'autres modèles qui capturent le comportement temporel des systèmes IdO. Cependant, il est encore nécessaire de développer des modèles et des techniques plus sophistiqués qui puissent gérer les interactions et les dépendances complexes qui existent dans les systèmes IdO. De plus, il est nécessaire de s'assurer que

les techniques de V&V sont intégrées dans le processus de développement logiciel des systèmes IdO, plutôt que d'être considérées comme une réflexion tardive. Cela nécessite un changement culturel vers une approche plus formalisée du développement logiciel, ainsi que le développement d'outils et de frameworks qui facilitent l'application des techniques de V&V.

Ainsi, les techniques de V&V offrent une approche prometteuse pour garantir la fiabilité et la sécurité des systèmes IdO. Cependant

8 Conclusion et travaux futurs

En conclusion, les techniques de FV&V ont le potentiel de relever les défis de la fiabilité et de la sécurité dans les systèmes IdO. Cependant, la nature dynamique et hétérogène de ces systèmes présente plusieurs défis pour l'application de ces techniques. Les chercheurs ont développé différentes techniques, telles que l'abstraction, la modularisation, la détection de symétrie et l'AT, pour relever ces défis. Cependant, il est nécessaire de développer des techniques plus efficaces et plus performantes capables de gérer les interactions complexes et les dépendances qui existent dans les systèmes IdO. De plus, il est nécessaire d'adopter une approche plus formalisée du développement logiciel, où les techniques de FV&V sont intégrées au processus de développement.

Une direction possible pour les futures recherches est le développement de modèles formels et de techniques de vérification plus sophistiqués capables de gérer la nature dynamique et hétérogène des systèmes IdO. Par exemple, les chercheurs pourraient développer des modèles qui capturent les interactions et les dépendances entre les appareils et les réseaux, ainsi que le contexte et l'environnement dans lesquels le système opère. Ils pourraient également développer des techniques de vérification capables de gérer le grand nombre d'états et d'événements qui se produisent dans les systèmes IdO, tout en maintenant les performances et la scalabilité. Une autre direction possible est le développement d'outils et de frameworks facilitant l'application des techniques de FV&V dans le processus de développement. Ces outils pourraient automatiser le processus de génération et de vérification des modèles, réduisant ainsi l'effort manuel requis et améliorant l'exactitude et la fiabilité des modèles.

De plus, il est nécessaire de développer des normes plus complètes pour les appareils et les réseaux IdO afin d'améliorer l'exactitude et la fiabilité des modèles formels. La normalisation peut également permettre l'interopérabilité entre les appareils et les réseaux, améliorant ainsi la scalabilité et la flexibilité des systèmes IdO. Enfin, il est nécessaire d'étudier l'utilisation de techniques d'apprentissage automatique et d'intelligence artificielle en conjonction avec les techniques de FV&V (45; 17; 24; 120). Ces techniques peuvent aider à identifier les motifs et les anomalies dans les systèmes IdO, améliorant ainsi leur fiabilité et leur sécurité.

En plus des orientations futures discutées ci-dessus, il y a deux autres orientations qui méritent d'être approfondies dans le contexte de la FV&V pour les systèmes IdO. Tout d'abord, l'intégration de nouvelles techniques d'IA dans les systèmes IdO dynamiques et hétérogènes est un domaine d'intérêt majeur. Des algorithmes d'IA spécifiques, tels que l'apprentissage fédéré et le 0NKP, ont montré des résultats prometteurs pour améliorer la sécurité des systèmes IdO. Les chercheurs pourraient explorer l'in-

tégration de ces techniques dans les modèles formels et les techniques de vérification pour détecter les vulnérabilités et garantir la sûreté et la sécurité des systèmes IdO.

Deuxièmement, il est nécessaire de fournir plus de détails sur la sécurité post-quantique (112). Avec l'impact potentiel de l'informatique quantique sur la sécurité des systèmes IdO, il est crucial d'étudier les mécanismes et les protocoles de sécurité post-quantique. Les chercheurs pourraient explorer l'utilisation de méthodes formelles en conjonction avec ces techniques pour garantir la sûreté et la sécurité des systèmes IdO face aux menaces émergentes.

En gardant ces orientations futures à l'esprit, la poursuite de la recherche et du développement des techniques de FV&V pour les systèmes IdO nécessitera une collaboration entre chercheurs, développeurs et acteurs de l'industrie. En abordant les défiset les problèmes ouverts discutés dans cet article, les techniques de FV&V peuvent contribuer à assurer le fonctionnement fiable et sécurisé des systèmes IdO, permettant ainsi de réaliser pleinement leur potentiel.

En résumé, les techniques de FV&V offrent une approche prometteuse pour garantir la fiabilité et la sécurité des systèmes IdO. Pour relever les défis et les problèmes ouverts discutés dans cet article, il faudra un effort concerté de la part des chercheurs, des développeurs et des acteurs de l'industrie. Grâce à la poursuite de la recherche et du développement, les techniques de FV&V peuvent contribuer à réaliser pleinement le potentiel des systèmes IdO de manière sûre et sécurisée.

Références

[1] Muhammad Abbas, Ali Hamayouni, Mahshid H Moghadam, Mehrdad Saadatmand, and Per E Strandberg. Making sense of failure logs in an industrial devops environment. In *International Conference on Information Technology-New Generations*, pages 217–226. Springer, 2023.

[2] Mohamed S Abdalzaher, Hussein A Elsayed, and Mostafa M Fouda. Employing remote sensing, data communication networks, ai, and optimization methodologies in seismology. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15 :9417–9438, 2022.

[3] Mohamed S Abdalzaher, Mostafa M Fouda, Hussein A Elsayed, and Mahmoud M Salim. Toward secured iot-based smart systems using machine learning. *IEEE Access*, 11 :20827–20841, 2023.

[4] Mohamed S Abdalzaher, Mahmoud M Salim, Hussein A Elsayed, and Mostafa M Fouda. Machine learning benchmarking for secured iot smart systems. In *2022 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)*, pages 50–56. IEEE, 2022.

[5] Mohamed S Abdalzaher, Lotfy Samy, and Osamu Muta. Non-zero-sum game-based trust model to enhance wireless sensor networks security for iot applications. *IET Wireless Sensor Systems*, 9(4) :218–226, 2019.

[6] Mohamed S Abdalzaher, M Sami Soliman, Sherif M El-Hady, Abderrahim Benslimane, and Mohamed Elwekeil. A deep learning model for earthquake parameters

observation in iot system-based earthquake early warning. *IEEE Internet of Things Journal*, 9(11) :8412–8424, 2021.

[7] Tanwir Ahmad, Junaid Iqbal, Adnan Ashraf, Dragos Truscan, and Ivan Porres. Model-based testing using uml activity diagrams : A systematic mapping study. *Computer Science Review*, 33 :98–112, 2019.

[8] Waqas Ahmad, Aamir Rasool, Abdul Rehman Javed, Thar Baker, and Zunera Jalil. Cyber security in iot-based cloud computing : A comprehensive survey. *Electronics*, 11(1) :16, 2021.

[9] Abdelmuttlib Ibrahim Abdalla Ahmed, Siti Hafizah Ab Hamid, Abdullah Gani, Ahmed Abdelaziz, and Mohammed Abaker. Formal analysis of trust and reputation for service composition in iot. *Sensors*, 23(6) :3192, 2023.

[10] Maryam Alamer and Mohammed Amin Almaiah. Cybersecurity in smart city : A systematic mapping study. In *2021 International Conference on Information Technology (ICIT)*, pages 719–724. IEEE, 2021.

[11] Amira Ali, Huda Amin Maghawry, and Nagwa Badr. Performance testing as a service using cloud computing environment : A survey. *Journal of Software : Evolution and Process*, page e2492, 2022.

[12] Omar Azib Alkhudaydi, Moez Krichen, and Ans D Alghamdi. A deep learning methodology for predicting cybersecurity attacks on the internet of things. *Information*, 14(10) :550, 2023.

[13] Azza Allouch, Omar Cheikhrouhou, Anis Koubâa, Khalifa Toumi, Mohamed Khalgui, and Tuan Nguyen Gia. Utm-chain : blockchain-based secure unmanned traffic management for internet of drones. *Sensors*, 21(9) :3049, 2021.

[14] Hamoud Alshammari, Karim Gasmi, Ibtihel Ben Ltaifa, Moez Krichen, Lassaad Ben Ammar, and Mahmood A Mahmood. Olive disease classification based on vision transformer and cnn models. *Computational Intelligence and Neuroscience*, 2022, 2022.

[15] Hashem Alyami, Wael Alosaimi, Moez Krichen, and Roobaea Alroobaea. Monitoring social distancing using artificial intelligence for fighting covid-19 virus spread. *International Journal of Open Source Software and Processes (IJOSSP)*, 12(3) :48–63, 2021.

[16] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the mirai botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1093–1110, Vancouver, BC, 2017. USENIX Association.

[17] Rubby Aworka, Lontsi Saadio Cedric, Wilfried Yves Hamilton Adoni, Jérémie Thouakesseh Zoueu, Franck Kalala Mutombo, Charles Lebon Mberi Kimpolo, Tarik Nahhal, and Moez Krichen. Agricultural decision system based on advanced machine learning models for yield prediction : Case of east african countries. *Smart Agricultural Technology*, 2 :100048, 2022.

[18] Benjamin Aziz. A formal model and analysis of an iot protocol. *Ad Hoc Networks*, 36 :49–57, 2016.

[19] Bojana Bakić, Miloš Milić, Ilija Antović, Dušan Savić, and Tatjana Stojanović. 10 years since stuxnet : What have we learned from this mysterious computer software worm ? In *2021 25th International Conference on Information Technology (IT)*, pages 1–4. IEEE, 2021.

[20] Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of model checking*, pages 305–343. Springer, 2018.

[21] Saddek Bensalem, Moez Krichen, Lotfi Majdoub, Riadh Robbana, and Stavros Tripakis. A simplified approach for testing real-time systems based on action refinement. In *ISoLA*, pages 191–202, 2007.

[22] Wadii Boulila, Maha Driss, Eman Alshanqiti, Mohamed Al-Sarem, Faisal Saeed, and Moez Krichen. Weight initialization techniques for deep learning algorithms in remote sensing : Recent trends and future perspectives. *Advances on Smart and Soft Computing : Proceedings of ICACIn 2021*, pages 477–484, 2022.

[23] Jerry R Burch, Roberto Passerone, and Alberto L Sangiovanni-Vincentelli. Modeling techniques in design-by-refinement methodologies. In *System Specification & Design Languages*, pages 283–292. Springer, 2003.

[24] Lontsi Saadio Cedric, Wilfried Yves Hamilton Adoni, Rubby Aworka, Jérémie Thouakesseh Zoueu, Franck Kalala Mutombo, Moez Krichen, and Charles Lebon Mberi Kimpolo. Crops yield prediction based on machine learning models : Case of west african countries. *Smart Agricultural Technology*, 2 :100049, 2022.

[25] Omar Cheikhrouhou and Anis Koubâa. Blockloc : Secure localization in the internet of things using blockchain. In *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 629–634. IEEE, 2019.

[26] Patrick Cousot. Abstract interpretation based formal methods and future challenges. In *Informatics*, pages 138–156. Springer, 2001.

[27] Jennifer A Davis, Matthew Clark, Darren Cofer, Aaron Fifarek, Jacob Hinchman, Jonathan Hoffman, Brian Hulbert, Steven P Miller, and Lucas Wagner. Study on the barriers to the industrial adoption of formal methods. In *Int. workshop on formal methods for industrial critical systems*, pages 63–77. Springer, 2013.

[28] Steve Easterbrook and John Callahan. Formal methods for verification and validation of partial specifications : A case study. *Journal of Systems and Software*, 40(3) :199–210, 1998.

[29] Eman K Elsayed, LS Diab, and Asmaa A Ibrahim. Formal verification of an efficient architecture to enhance the security in iot. *International Journal of Advanced Computer Science and Applications*, 12(3), 2021.

[30] Abdelouahab Fortas, Elhillali Kerkouche, and Allaoua Chaoui. Formal verification of iot applications using rewriting logic : An mde-based approach. *Science of Computer Programming*, 222 :102859, 2022.

[31] Ying Fu, Meng Ren, Fuchen Ma, Heyuan Shi, Xin Yang, Yu Jiang, Huizhong Li, and Xiang Shi. Evmfuzzer : detect evm vulnerabilities via fuzz testing. In *Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pages 1110–1114, 2019.

[32] Jie Gao and Muthu Ramachandran. A survey on software testing techniques using artificial intelligence. *Journal of Big Data*, 5(1) :1–35, 2018.

[33] Vahid Garousi, Alper Buğra Keleş, Yunus Balaman, Zeynep Özdemir Güler, and Andrea Arcuri. Model-based testing in practice : An experience report from the web applications domain. *Journal of Systems and Software*, 180 :111032, 2021.

[34] Herman Geuvers. Proof assistants : History, ideas and future. *Sadhana*, 34(1) :3–25, 2009.

[35] Mario Gleirscher, Simon Foster, and Jim Woodcock. New opportunities for integrated formal methods. *ACM Computing Surveys (CSUR)*, 52(6) :1–36, 2019.

[36] Mario Gleirscher and Diego Marmsoler. Formal methods in dependable systems engineering : a survey of professionals from europe and north america. *Empirical Software Engineering*, 25(6) :4473–4546, 2020.

[37] Anjana Gosain and Ganga Sharma. Static analysis : A survey of techniques and tools. In *Intelligent Computing and Applications*. Springer, 2015.

[38] Manik Gupta, Rakesh Kumar, Shashi Shekhar, Bhisham Sharma, Ram Bahadur Patel, Shaily Jain, Imed Ben Dhaou, and Celestine Iwendi. Game theory-based authentication framework to secure internet of vehicles with blockchain. *Sensors*, 22(14) :5119, 2022.

[39] Jon Hagar and Marc-Florian Wendland. Defining software test architectures with the uml testing profile. In *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 271–280. IEEE, 2023.

[40] Fatima Hassan, Syed Fawad Hussain, and Saeed Mian Qaisar. Fusion of multivariate eeg signals for schizophrenia detection using cnn and machine learning techniques. *Information Fusion*, 92 :466–478, 2023.

[41] Morten Hertzum. Usability testing : A practitioner's guide to evaluating the user experience. *Synthesis Lectures on Human-Centered Informatics*, 13(1) :i–105, 2020.

[42] Katharina Hofer-Schmitz and Branka Stojanović. Towards formal methods of iot application layer protocols. In *2019 12th CMI conference on cybersecurity and privacy (CMI)*, pages 1–6. IEEE, 2019.

[43] Katharina Hofer-Schmitz and Branka Stojanović. Towards formal verification of iot protocols : A review. *Computer Networks*, 174 :107233, 2020.

[44] Andreas Holzinger, Anna Saranti, Alessa Angerschmid, Carl Orge Retzlaff, Andreas Gronauer, Vladimir Pejakovic, Francisco Medel-Jimenez, Theresa Krexner, Christoph Gollob, and Karl Stampfer. Digital transformation in smart farm and forest operations needs human-centered ai : Challenges and future directions.

Sensors, 22(8) :3043, 2022.

[45] Olfa Hrizi, Karim Gasmi, Ibtihel Ben Ltaifa, Hamoud Alshammari, Hanen Karamti, Moez Krichen, Lassaad Ben Ammar, and Mahmood A Mahmood. Tuberculosis disease diagnosis based on an optimized machine learning model. *Journal of Healthcare Engineering*, 2022, 2022.

[46] Syed Ibrahim Imtiaz, Liaqat Ali Khan, Ahmad S Almadhor, Sidra Abbas, Shtwai Alsubai, Michal Gregus, and Zunera Jalil. Efficient approach for anomaly detection in internet of things traffic using deep learning. *Wireless Communications & Mobile Computing*, 2022.

[47] Rateb Jabbar, Moez Krichen, Noora Fetais, and Kamel Barkaoui. Adopting formal verification and model-based testing techniques for validating a blockchain-based healthcare records sharing system. In *22nd International Conference on Enterprise Information Systems*, pages 261–268. SCITEPRESS-Science and Technology Publications, 2020.

[48] Rateb Jabbar, Moez Krichen, Mohamed Kharbeche, Noora Fetais, and Kamel Barkaoui. A formal model-based testing framework for validating an iot solution for blockchain-based vehicles communication. In *15th International Conference on Evaluation of Novel Approaches to Software Engineering*, pages 595–602. SCITEPRESS-Science and Technology Publications, 2020.

[49] Rateb Jabbar, Mohammed Shinoy, Mohamed Kharbeche, Khalifa Al-Khalifa, Moez Krichen, and Kamel Barkaoui. Urban traffic monitoring and modeling system : An iot solution for enhancing road safety. In *2019 international conference on internet of things, embedded systems and communications (iintec)*, pages 13–18. IEEE, 2019.

[50] Abdul Rehman Javed, Faisal Shahzad, Saif ur Rehman, Yousaf Bin Zikria, Imran Razzak, Zunera Jalil, and Guandong Xu. Future smart cities : Requirements, emerging technologies, applications, challenges, and future aspects. *Cities*, 129 :103794, 2022.

[51] Bryer Jeannotte and Ali Tekeoglu. Artorias : Iot security testing framework. In *2019 26th International Conference on Telecommunications (ICT)*, pages 233–237. IEEE, 2019.

[52] K Keerthi, Indrani Roy, Aritra Hazra, and Chester Rebeiro. Formal verification for security in iot devices. *Security and Fault Tolerance in Internet of Things*, pages 179–200, 2019.

[53] Amleset Kelati, Imed Ben Dhaou, and Hannu Tenhunen. Biosignal monitoring platform using wearable iot. In *Proceedings of the 22st Conference of Open Innovations Association FRUCT*, pages 332–337, 2018.

[54] Vladimir Khorikov. *Unit Testing Principles, Practices, and Patterns*. Simon and Schuster, 2020.

[55] Aron Kondoro, Imed Ben Dhaou, Hannu Tenhunen, and Nerey Mvungi. Real time performance analysis of secure iot protocols for microgrid communication. *Future Generation Computer Systems*, 116 :1–12, 2021.

[56] Anis Koubaa, Azza Allouche, Mohamed Khalgui, and Omar Cheikhrouhou. Blockchain-based solution for internet of drones security and privacy, November 1 2022. US Patent 11,488,488.

[57] Moez Krichen. *Model-based testing for real-time systems*. PhD thesis, PhD thesis, PhD thesis, Universit Joseph Fourier (December 2007), 2007.

[58] Moez Krichen. A formal framework for conformance testing of distributed real-time systems. In *International Conference On Principles Of Distributed Systems*, pages 139–142. Springer, 2010.

[59] Moez Krichen. A formal framework for black-box conformance testing of distributed real-time systems. *International Journal of Critical Computer-Based Systems*, 3(1-2) :26–43, 2012.

[60] Moez Krichen. *Contributions to model-based testing of dynamic and distributed real-time systems*. PhD thesis, École Nationale d'Ingénieurs de Sfax (Tunisie), 2018.

[61] Moez Krichen. Anomalies detection through smartphone sensors : A review. *IEEE Sensors Journal*, 21(6) :7207–7217, 2021.

[62] Moez Krichen. Convolutional neural networks : A survey. *Computers*, 12(8) :151, 2023.

[63] Moez Krichen. Deep reinforcement learning. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2023.

[64] Moez Krichen. Formal methods and validation techniques for ensuring automotive systems security. *Information*, 14(12) :666, 2023.

[65] Moez Krichen. Generative adversarial networks. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pages 1–7. IEEE, 2023.

[66] Moez Krichen. A survey on formal verification and validation techniques for internet of things. *Applied Sciences*, 13(14) :8122, 2023.

[67] Moez Krichen, Wilfried Yves Hamilton Adoni, Alaeddine Mihoub, Mohammed Y Alzahrani, and Tarik Nahhal. Security challenges for drone communications : Possible threats, attacks and countermeasures. In *2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, pages 184–189. IEEE, 2022.

[68] Moez Krichen and Roobaea Alroobaea. Towards optimizing the placement of security testing components for internet of things architectures. In *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, pages 1–2. IEEE, 2019.

[69] Moez Krichen, Omar Cheikhrouhou, Mariam Lahami, Roobaea Alroobaea, and Afef Jmal Maâlej. Towards a model-based testing framework for the security of internet of things for smart city applications. In *Smart Societies, Infrastructure, Technologies and Applications : First International Conference, SCITA 2017, Jeddah, Saudi Arabia, November 27–29, 2017, Proceedings 1*, pages 360–365.

Springer International Publishing, 2018.

[70] Moez Krichen and Mariam Lahami. Towards a runtime testing framework for dynamically adaptable internet of things networks in smart cities. *Smart Infrastructure and Applications : Foundations for Smarter Cities and Societies*, pages 589–607, 2020.

[71] Moez Krichen, Mariam Lahami, Omar Cheikhrouhou, Roobaea Alroobaea, and Afef Jmal Maâlej. Security testing of internet of things for smart city applications : A formal approach. In *Smart Infrastructure and Applications*, pages 629–653. Springer, Cham, 2020.

[72] Moez Krichen, Afef Jmal Maâlej, and Mariam Lahami. A model-based approach to combine conformance and load tests : an ehealth case study. *International Journal of Critical Computer-Based Systems*, 8(3-4) :282–310, 2018.

[73] Moez Krichen, Seifeddine Mechti, Roobaea Alroobaea, Elyes Said, Parminder Singh, Osamah Ibrahim Khalaf, and Mehedi Masud. A formal testing model for operating room control system using internet of things. *Computers, Materials & Continua*, 66(3) :2997–3011, 2021.

[74] Moez Krichen and Alaeddine Mihoub. Unmanned aerial vehicles communications security challenges : A survey. In *Unmanned Aerial Vehicles Applications : Challenges and Trends*, pages 349–373. Springer International Publishing Cham, 2023.

[75] Moez Krichen, Alaeddine Mihoub, Mohammed Y Alzahrani, Wilfried Yves Hamilton Adoni, and Tarik Nahhal. Are formal methods applicable to machine learning and artificial intelligence ? In *2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, pages 48–53. IEEE, 2022.

[76] Moez Krichen and Stavros Tripakis. Real-time testing with timed automata testers and coverage criteria. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 134–151. Springer, Berlin, Heidelberg, 2004.

[77] Moez Krichen and Stavros Tripakis. State identification problems for timed automata. In *Testing of Communicating Systems : 17th IFIP TC6/WG 6.1 International Conference, TestCom 2005, Montreal, Canada, May 31-June, 2005. Proceedings 17*, pages 175–191. Springer Berlin Heidelberg, 2005.

[78] Moez Krichen and Stavros Tripakis. Interesting properties of the real-time conformance relation tioco. In *International Colloquium on Theoretical Aspects of Computing*, pages 317–331. Springer Berlin Heidelberg Berlin, Heidelberg, 2006.

[79] Asif Ali Laghari, Kaishan Wu, Rashid Ali Laghari, Mureed Ali, and Abdullah Ayub Khan. A review and state of art of internet of things (iot). *Archives of Computational Methods in Engineering*, pages 1–19, 2021.

[80] Mariam Lahami, Fairouz Fakhfakh, Moez Krichen, and Mohamed Jmaiel. Towards a ttcn-3 test system for runtime testing of adaptable and distributed systems. In *Testing Software and Systems : 24th IFIP WG 6.1 International Conference, ICTSS 2012, Aalborg, Denmark, November 19-21, 2012. Proceedings 24*,

pages 71–86. Springer Berlin Heidelberg, 2012.

[81] Mariam Lahami and Moez Krichen. A survey on runtime testing of dynamically adaptable and distributed systems. *Software Quality Journal*, 29(2) :555–593, 2021.

[82] Mariam Lahami, Moez Krichen, Hajer Barhoumi, and Mohamed Jmaiel. Selective test generation approach for testing dynamic behavioral adaptations. In *IFIP International Conference on Testing Software and Systems*, pages 224–239. Springer, Cham, 2015.

[83] Mariam Lahami, Moez Krichen, Mariam Bouchakwa, and Mohamed Jmaiel. Using knapsack problem model to design a resource aware test architecture for adaptable and distributed systems. In *Testing Software and Systems : 24th IFIP WG 6.1 International Conference, ICTSS 2012, Aalborg, Denmark, November 19-21, 2012. Proceedings 24*, pages 103–118. Springer Berlin Heidelberg, 2012.

[84] Mariam Lahami, Moez Krichen, and Mohamed Jmaïel. Runtime testing approach of structural adaptations for dynamic and distributed systems. *International Journal of Computer Applications in Technology*, 51(4) :259–272, 2015.

[85] In Lee. Internet of things (iot) cybersecurity : Literature review and iot cyber risk management. *Future Internet*, 12(9) :157, 2020.

[86] Afef Jmal Maâlej, Manel Hamza, Moez Krichen, and Mohamed Jmaiel. Automated significant load testing for ws-bpel compositions. In *2013 IEEE sixth international conference on software testing, verification and validation workshops*, pages 144–153. IEEE, 2013.

[87] Afef Jmal Maâlej and Moez Krichen. Study on the limitations of ws-bpel compositions under load conditions. *The Computer Journal*, 58(3) :385–402, 2015.

[88] Afef Jmal Maâlej and Moez Krichen. A model based approach to combine load and functional tests for service oriented architectures. In *VECoS*, pages 123–140, 2016.

[89] Afef Jmal Maâlej, Moez Krichen, and Mohamed Jmaiel. Conformance testing of ws-bpel compositions under various load conditions. In *2012 IEEE 36th annual computer software and applications conference*, pages 371–371. IEEE, 2012.

[90] Afef Jmal Maâlej, Moez Krichen, and Mohamed Jmaiel. Model-based conformance testing of ws-bpel compositions. In *2012 IEEE 36th annual computer software and applications conference workshops*, pages 452–457. IEEE, 2012.

[91] Afef Jmal Maâlej, Mariam Lahami, Moez Krichen, and Mohamed Jmaïel. Distributed and resource-aware load testing of ws-bpel compositions. In *ICEIS (2)*, pages 29–38, 2018.

[92] Asmaa Maher, Saeed Mian Qaisar, N Salankar, Feng Jiang, Ryszard Tadeusiewicz, Paweł Pławiak, Ahmed A Abd El-Latif, and Mohamed Hammad. Hybrid eeg-fnirs brain-computer interface based on the non-linear features extraction and stacking ensemble learning. *biocybernetics and biomedical engineering*, 43(2) :463–475, 2023.

[93] Abdul Malik, Muhammad Zahid Khan, Saeed Mian Qaisar, Mohammad Faisal,

and Gulzar Mehmood. An efficient approach for the detection and prevention of gray-hole attacks in vanets. *IEEE Access*, 2023.

[94] Sara N Matheu-García, José L Hernández-Ramos, Antonio F Skarmeta, and Gianmarco Baldini. Risk-based automated assessment and testing for the cybersecurity certification and labelling of iot devices. *Computer Standards & Interfaces*, 62 :64–83, 2019.

[95] Saeed Mian Qaisar, Dalila Say, Salah Zidi, and Krichen Moez. Automated categorization of multiclass welding defects using the x-ray image augmentation and convolutional neural network. 2023.

[96] Franc Mihalič, Mitja Truntič, and Alenka Hren. Hardware-in-the-loop simulations : A historical overview of engineering challenges. *Electronics*, 11(15) :2462, 2022.

[97] Alaeddine Mihoub. A deep learning-based framework for human activity recognition in smart homes. *Mobile Information Systems*, 2021 :1–11, 2021.

[98] Alaeddine Mihoub, Moez Krichen, Mohannad Alswailim, Sami Mahfoudhi, and Riadh Bel Hadj Salah. Road scanner : A road state scanning approach based on machine learning techniques. *Applied Sciences*, 13(2) :683, 2023.

[99] Alaeddine Mihoub and Grégoire Lefebvre. Social intelligence modeling using wearable devices. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, pages 331–341, 2017.

[100] Barton P Miller, Mengxiao Zhang, and Elisa R Heymann. The relevance of classic fuzz testing : Have we solved this one ? *IEEE Transactions on Software Engineering*, 48(6) :2028–2039, 2020.

[101] Markus Müller-Olm, David Schmidt, and Bernhard Steffen. Model-checking. In *International Static Analysis Symposium*, pages 330–354. Springer, 1999.

[102] Hussam Saeed Musa, Moez Krichen, Adem Alpaslan Altun, and Meryem Ammi. Survey on blockchain-based data storage security for android mobile applications. *Sensors*, 23(21) :8749, 2023.

[103] Muneeba Nasir, Abdul Rehman Javed, Muhammad Adnan Tariq, Muhammad Asim, and Thar Baker. Feature engineering and deep learning-based intrusion detection framework for securing edge iot. *The Journal of Supercomputing*, pages 1–15, 2022.

[104] Dhouha Ben Noureddine, Moez Krichen, Seifeddine Mechti, Tarik Nahhal, and Wilfried Yves Hamilton Adoni. An agent-based architecture using deep reinforcement learning for the intelligent internet of things applications. In *Advances on Smart and Soft Computing : Proceedings of ICACIn 2020*, pages 273–283. Springer Singapore, 2021.

[105] Amir Pnueli, Sitvanit Ruah, and Lenore Zuck. Automatic deductive verification with invisible invariants. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 82–97. Springer, 2001.

[106] Mehrdad Saadatmand, Eduard Paul Enoiu, Holger Schlingloff, Michael Felderer, and Wasif Afzal. Smartdelta : automated quality assurance and optimization in

incremental industrial software systems development. In *2022 25th Euromicro Conference on Digital System Design (DSD)*, pages 754–760. IEEE, 2022.

[107] S Phani Shashank, Praneeth Chakka, and D Vijay Kumar. A systematic literature survey of integration testing in component-based software engineering. In *2010 International Conference on Computer and Communication Technology (ICCCT)*, pages 562–568. IEEE, 2010.

[108] Min-Zheng Shieh, Yi-Bing Lin, and Yin-Jui Hsu. Verificationtalk : A verification and security mechanism for iot applications. *Sensors*, 21(22) :7449, 2021.

[109] Shachar Siboni, Vinay Sachidananda, Yair Meidan, Michael Bohadana, Yael Mathov, Suhas Bhairav, Asaf Shabtai, and Yuval Elovici. Security testbed for internet-of-things devices. *IEEE transactions on reliability*, 68(1) :23–44, 2019.

[110] Marir Souad, Belala Faiza, and Hameurlain Nabil. Formal modeling iot systems on the basis of biagents* and maude. In *2020 International Conference on Advanced Aspects of Software Engineering (ICAASE)*, pages 1–7. IEEE, 2020.

[111] Alireza Souri and Monire Norouzi. A state-of-the-art survey on formal verification of the internet of things applications. *Journal of Service Science Research*, 11(1) :47–67, 2019.

[112] Dan-Alexandru Teodoraș, Eduard-Cristian Popovici, George Suciu, and Mari-Anais Sachian. Quantum technology's role in cybersecurity. In *Advanced Topics in Optoelectronics, Microelectronics, and Nanotechnologies XI*, volume 12493, pages 96–103. SPIE, 2023.

[113] Jing Tian, Yuanfang Li, and Xiaoyuan Zhang. A survey on software testing with machine learning. *Journal of Software : Evolution and Process*, 31(7) :e2176, 2019.

[114] Zinah Hussein Toman, Lazhar Hamel, Sarah Hussein Toman, Mohamed Graiet, and Dalton Cézane Gomes Valadares. Formal verification for security and attacks in iot physical layer. *Journal of Reliable Intelligent Environments*, pages 1–19, 2023.

[115] Porfirio Tramontana, Domenico Amalfitano, Nicola Amatucci, and Anna Rita Fasolino. Automated functional testing of mobile applications : a systematic mapping study. *Software Quality Journal*, 27(1) :149–201, 2019.

[116] Jack van Heugten Breurkes, Fabian Gilson, and Matthias Galster. Overlap between automated unit and acceptance testing–a systematic literature review. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, pages 80–89, 2022.

[117] Juan Pablo Vielma. Mixed integer linear programming formulation techniques. *Siam Review*, 57(1) :3–57, 2015.

[118] Fujun Wang, Zining Cao, Lixing Tan, and Hui Zong. Survey on learning-based formal methods : Taxonomy, applications and possible future directions. *IEEE Access*, 8 :108561–108578, 2020.

[119] Bin Xie, Shuai Wang, Xiuheng Wu, Changkai Wen, Shengli Zhang, and Xueyan Zhao. Design and hardware-in-the-loop test of a coupled drive system for electric

tractor. *Biosystems Engineering*, 216 :165–185, 2022.

[120] Salah Zidi, Alaeddine Mihoub, Saeed Mian Qaisar, Moez Krichen, and Qasem Abu Al-Haija. Theft detection dataset for benchmarking and machine learning based classification in a smart grid environment. *Journal of King Saud University-Computer and Information Sciences*, 35(1) :13–25, 2023.