Conduire des sessions de Test Exploratoire, une approche basée sur les données.

Julien Leveau

université de BORDEAUX

THÈSE

Présentée au Laboratoire Bordelais de Recherche en Informatique pour obtenir le grade de Docteur de l'Université de Bordeaux

Spécialité	:	**Informatique**
Formation Doctorale	:	**Informatique**
École Doctorale	:	**Mathématiques et Informatique**

Conduire des sessions de Test Exploratoire, une approche basée sur les données

Par

Julien Leveau

Sous la direction de Pr. Xavier Blanc
et de Pr. Laurent Réveillère

Soutenue le 19/11/2021

Membres du jury :

Mme. DUCHIEN, Laurence	Professeur, Université de Lille.	Présidente
M. BISSYANDE, Tegawendé François,	Professeur, Université de Luxembourg.	Rapporteur
M. LEGEARD, Bruno	Professeur, Université de Bourgogne Franche Comté.	Rapporteur
M. BLANC, Xavier	Professeur, Université de Bordeaux.	Directeur
M. REVEILLERE, Laurent	Professeur, Université de Bordeaux.	Directeur

---

Titre : Assister les Sessions de testEexploratoire, une Approche basée sur les Données.

Résumé : Les systèmes logiciels, notamment les applications web, jouent un rôle majeur dans notre vie personnelle et professionnelle. Il est essentiel de minimiser les défaillances, mais en même temps, ils deviennent de plus en plus complexes et donc difficiles à tester. Les tests exploratoires se sont révélés être une méthode efficace pour trouver les bugs qui nécessitent des interactions complexes avec le système. Ils s'appuient sur les connaissances métier et l'expérience des testeurs pour valider le système. Cependant, l'outillage des testeurs est sous-développé dans ce domaine. Nous affirmons que le support des tests exploratoires peut améliorer la qualité des tests et diminuer le niveau d'expertise requis pour les testeurs. Les connaissances métier aident les testeurs à identifier les domaines d'intérêt dans le système et l'expérience des testeurs les aide à maintenir de la diversité dans leurs tests. Dans cette thèse, nous proposons et évaluons des approches pour aider les testeurs à effectuer des tests exploratoires efficaces. Nous proposons de modéliser les interactions réalisées par les testeurs exploratoires, afin de recommander des actions d'intérêt, et de fournir un retour direct pendant la session. Nous basons nos recommandations sur des modèles de Markov et des machines à états. Pour évaluer nos approches, nous avons développé une base logicielle. Nous espérons que ce logiciel servira de base à de futurs travaux basés sur les interactions avec les pages web. Nos résultats montrent que les approches proposées aident les testeurs sur des applications web réelles. Notre objectif pour les travaux futurs est d'améliorer les conseils proposés, en apportant plus d'automatisation dans les tests exploratoires.

Mots clés :
Test des logiciels
Test exploratoire
Applications web

Title : Assisting exploratory testing sessions, a data-driven approach.

Abstract : Software systems, especially web applications, play a major role in our personal and professional lives. It is essential to minimize failures, but at the same time, they are becoming more and more complex and therefore difficult to test. Exploratory testing has proven to be an efficient method to find bugs that require complex interactions with the system. They rely on the business knowledge and experience of testers to validate the system. However, tester support is underdeveloped in this area. We claim that exploratory test support can improve test quality and decrease the level of expertise required for testers. Business knowledge helps

testers identify areas of interest in the system, and the testing experience helps the testers to maintain the diversity level of the tests they perform. In this thesis, we propose and evaluate approaches to help testers perform effective exploratory testing. We propose to model the interactions performed by exploratory testers, recommend actions of interest, and provide direct feedback during the session. We base our recommendations on Markov models and state machines. To evaluate our approaches, we have developed a software base. We hope that this software will serve as a basis for future work based on interactions with web pages. Our results show that the proposed approaches help testers on real web applications. Our goal for future work is to improve the proposed guidance, bringing more automation in exploratory testing.

Keywords :
software testing
Exploratory Testing
Web Applications

Laboratoire Bordelais de Recherche en Informatique (LaBRI). Université de Bordeaux.
351, cours de la Libération – 33 405 TALENCE.
&
CIS Valley. 3 Rue Adrienne Bolland – 33185 Le Haillan.

Table des matières

Chapitre 1

Introduction

Sommaire

. Ce chapitre d'introduction présente le contexte de cette thèse CIFRE, réalisé avec la société CIS Valley.

1.1 Context

L'utilisation des services numériques a considérablement augmenté depuis le début des années 1990 et le cap du milliard de sites web a été franchi en 2014 [1]. Ces derniers jouent d'ailleurs un rôle majeur dans notre vie, que ce soit au niveau personnel, aussi bien que professionnel. De ce fait, le dysfonctionnement d'un service peut entraîner des désagréments importants pour les utilisateurs et des pertes financières pour les entreprises qui les proposent.

Les applications web sont une composition de plusieurs services, parfois développés et maintenus par différents acteurs. Face à la complexité croissante de ces sys-

tèmes, les efforts des acteurs pour assurer la qualité des outils logiciels et en réduire le coût, s'orientent vers l'automatisation. Les actions répétitives telles que les tests de non-régression exécutés régulièrement peuvent être automatisées. Les méthodes de gestion de projet agile sont devenues les principales techniques d'organisation dans le secteur des applications web et, selon le rapport 2020 State of Agile [3], 95 % des entreprises déclarent les utiliser. Elles sont basées sur des cycles de développement courts et accordent une grande importance aux tests automatisés, principalement au niveau unitaire. La rapidité des cycles de développement ne laisse pas le temps de construire et maintenir à jour une spécification complète du système.

De plus, il est fréquent qu'une application soit développée à l'aide de bibliothèques externes, de frameworks ou de services web développés par d'autres équipes. Une équipe de développement travaillant sur un service web n'a de visibilité que sur le code qu'elle produit et considère les autres composants du système comme des boîtes noires. Lorsque le code des autres composants n'est pas disponible, il n'est alors pas possible de construire des tests en boîte blanche au niveau du système, il faut donc tester le système en boite noire.

Notre partenaire industriel CIS Valley emploie 160 personnes dans 5 agences. L'entreprise, entre autres activités, développe SONATE, un ERP (*Enterprise Resource Planning*) principalement utilisé par les organisations publiques. Le développement de SONATE a débuté en 2017 et son coup a été estimé à 3000 jours-homme. SONATE est une application web composée d'un frontend, basé sur le framework Angular, et d'un backend écrit en C#.

Le projet implique trois équipes (équipes de fonctionnalités, de développement et de migration) supervisées par deux managers et un directeur de projet. L'équipe chargée des fonctionnalités est composée de trois consultants commerciaux seniors qui sont chargés des aspects fonctionnels de SONATE. L'équipe fonctionnelle rédige manuellement les scénarios de test de validation, qui sont exécutés manuellement pendant la phase de recette interne. L'équipe de développement comprend quatre développeurs : un développeur front-end et trois développeurs back-end. Trois développeurs ont plus

de cinq ans d'expérience, le dernier est un développeur junior. Les équipes de développement cultivent une culture de test, que ce soit par la couverture du code à l'aide de tests unitaires ou par l'écriture manuelle de tests automatisés de bout en bout. L'équipe de migration est composée de quatre membres seniors qui sont chargés de transférer et de synchroniser la version précédente de SONATE avec la version plus récente.

La nature de SONATE implique de nombreuses règles métier qui changent fréquemment, ce qui implique des changements importants dans la base de code. Comme les scénarios de test sont écrits manuellement ces changements rendent difficile le maintien d'une base de scénarios de test. Pour cette raison, CIS Valley est intéressé par l'application de tests exploratoires.

Le Test Exploratoire (TE) est une approche efficace pour le test en boîte noire des systèmes [4, 31]. Il donne au testeur la responsabilité de la conception du test qu'il exécute. Plusieurs études montrent que l'efficacité du TE est comparable aux tests scriptés, permettant de trouver des défauts demandant des séquences d'interactions complexes avec le système [4, 24, 29]. Itkonen et Rautiainen ont conduit une série d'entretiens avec des entreprises pratiquant le TE, toutes rapportent qu'il permet de tester plus en profondeur les fonctionnalités, incluant des tests qui n'auraient pas fait partie d'un plan de test, c'est par exemple le cas pour les dépendances entre d'anciennes et de nouvelles fonctionnalités qui n'auraient pas été testées en construisant des cas de test pour couvrir des exigences sur la nouvelle fonctionnalité. Ce qui est pourtant souvent le cas dans le contexte des applications web, faisant intervenir de nombreux composants. Nous considérons que le test exploratoire est une pratique importante pour la stratégie de test d'une équipe développant une application web. Cependant, la pratique du test exploratoire est encore peu outillée et demande aux testeurs un niveau d'expertise important pour être efficiente. Nous soutenons que les testeurs peuvent être guidés pendant leur exploration du logiciel. Nous proposons dans cette thèse des approches pour guider les testeurs dans le but d'améliorer l'efficacité du TE.

1.2 Objectifs

Nous abordons deux axes d'amélioration pour la conduite de tests exploratoires. Premièrement, le problème de la diversité fait référence à la couverture des actions et des combinaisons d'actions qui peuvent être effectuées sur l'application web. Il n'est pas possible de tester de façon exhaustive, mais l'augmentation du nombre de combinaisons réalisées par les testeurs est un indicateur que les tests ne se répètent pas. En d'autres termes, chaque nouveau test contribue à l'exploration du système.

Deuxièmement, le test exploratoire demande un haut niveau d'expertise de la part des testeurs. Un manque de connaissances peut conduire à des parties du système laissées inexplorées ou des sessions de test qui ne correspondent pas aux objectifs fixés. Nous proposons de guider ces testeurs qui ne connaissent pas l'application, pour leur permettre de collaborativement l'explorer. Nous illustrons cette approche à travers le crowdtesting, qui est une méthode permettant de faire participer des testeurs extérieurs au projet à des tâches de test

1.2.1 Contribution

Les travaux de cette thèse ont pour but d'améliorer l'efficacité des sessions de test exploratoire. Nous avons choisi de traiter ce problème en suivant deux axes :

Diversité : Améliorer le nombre de combinaisons d'actions réalisées.

Connaissance : Aider les testeurs à déterminer les objectifs des sessions de test exploratoire, lorsque ceux-ci connaissent mal le Système Testé (ST).

Nous proposons les contributions suivantes pour répondre à ces questions d'efficacité des sessions de tests exploratoires.

1. Un socle logiciel open source permettant d'assister les sessions de test exploratoire d'applications web. Il enregistre les actions des testeurs et les guide grâce à un retour visuel directement sur la page web. Les actions enregistrées peuvent être analysées pendant la session, afin de générer des recommandations pour le testeur.

Cet outil a pour objectif de servir de base à l'implémentation et l'évaluation de différentes techniques de recommandation.

2. Une approche qui s'efforce de favoriser la diversité des tests exploratoires. En particulier, nous abordons le cas difficile des applications web, car ces applications sont largement développées et déployées de nos jours. Pour atteindre cet objectif difficile, nous introduisons un modèle de prédiction basé sur des modèles de langage n-gram. Plus précisément, nous entraînons continuellement un modèle de prédiction en assimilant les traces de test précédemment effectuées par les testeurs. Ensuite, lorsqu'un testeur réalise un nouveau scénario de test, nous demandons au modèle de prédiction de proposer les prochaines interactions pertinentes pour le test, qui sont celles qui sont fréquemment réalisées. Une telle prédiction liste les interactions qui ont été fréquemment réalisées. Nous conseillons alors au testeur de ne pas suivre la prédiction, mais, au contraire, de considérer les interactions ayant un faible pourcentage de prédiction. Si le testeur suit notre conseil, il réalisera alors des interactions nouvelles ou rarement réalisées, ce qui augmentera la diversité du test. Une fois l'interaction réalisée, nous entraînons à nouveau le modèle de prédiction (avec la nouvelle trace du test), qui demandera alors plus de diversité pour les prochains tests.

3. Une approche pour organiser des sessions de test exploratoire en s'appuyant sur une plateforme de crowdsourcing. La difficulté d'utiliser une telle plateforme vient de l'expertise des crowdtesters et de leur capacité à explorer l'application web. Les crowdtesters ne sont pas des testeurs experts et des travaux antérieurs ont montré qu'ils ont des difficultés à suivre les instructions de test [20, 5]. Leur motivation principale est de maximiser leur revenu en accomplissant les tâches le plus rapidement possible, ce qui n'est pas l'état d'esprit du test exploratoire. En pratique, lorsqu'on fait appel à des crowdtesters, les chances que les testeurs restent dans le cadre de la session en cours sont très faibles. Heureusement, il a été démontré que les testeurs moins expérimentés, comme les crowdtesters, obtiennent de meilleurs

résultats lorsqu'ils sont guidés [45]. Notre idée principale est d'introduire une approche qui permet aux concepteurs de sessions de test exploratoire de modéliser formellement l'objectif de la session. Nous tirons ensuite parti de ce modèle pour fournir une orientation visuelle en direct aux crowdtesters qui mènent la session. En utilisant cette approche, nous réduisons significativement les chances que les crowdtesters ne suivent pas l'objectif de la session, sans sacrifier la diversité des explorations réalisées dans le cadre de l'objectif de la session.

1.3 Structure de la thèse

Le contenu de cette thèse est organisé comme suit :

Le Chapitre 2 présente une vue d'ensemble de l'état de l'art dans le domaine du test exploratoire des applications web. Le Chapitre 3 présente les services d'AIFEX, un outil qui implémente les approches décrites dans ces travaux. Le Chapitre 4 présente notre approche pour traiter de la diversité des sessions de test exploratoire. Le Chapitre 5 présente notre approche pour aider les testeurs à réaliser des tests pertinents. Le Chapitre 6 est une discussion sur l'application de nos contributions aux méthodes de test et nos futurs travaux.

Chapitre 2

État de l'art

Sommaire

Dans ce chapitre, nous rappelons les définitions des fondements nécessaires à notre travail. Ces principes sont présentés en trois parties.

Premièrement, nous définissons les principaux termes du test logiciel, en mettant l'accent sur le test manuel à travers le test exploratoire, les méthodes de crowdtes-

ting, et enfin les techniques existantes pour guider les testeurs lors des sessions de test manuel. Deuxièmement, nous présentons le fonctionnement des applications web et les techniques existantes pour suivre les actions des utilisateurs sur ces applications. Troisièmement, nous abordons les outils permettant de modéliser une application et le comportement des utilisateurs ou des testeurs des applications web. Nous présentons l'utilisation qui a été faite de ces modèles dans les techniques de test logiciel.

2.1 Test des logiciels

Le test est une vérification du système cruciale pour la qualité d'un logiciel. Le Comité international de qualification du test logiciel (ISTQB) définit le test comme :

Définition 2.1.1 (Test logiciel)**.** Un ensemble de cas à tester (état de l'objet à tester avant exécution du test, actions ou données en entrée, valeurs ou observations attendues, et état de l'objet après exécution), éventuellement accompagné d'une procédure d'exécution (séquence d'actions à exécuter). Il est lié à un objectif.

Le test d'un logiciel est une activité complexe qui nécessite différents niveaux d'abstraction allant des lignes de code, les interactions entre les composants logiciels, jusqu'aux fonctionnalités du système. Le but d'une pratique de test n'est pas directement de trouver des bugs, mais de couvrir des objectifs pour déterminer si un ensemble d'utilisations fonctionne correctement. Si le critère utilisé pour déterminer les objectifs est exigeant, et que tous les objectifs ont été couverts sans bugs, c'est un indicateur fort de la bonne qualité du logiciel.

2.1.1 Terminologie

Les définitions présentées dans cette section sont tirées du glossaire de l'ISTQB.

Les activités de test sont décomposées pour tester des sous-parties spécifiques du système. Dans cette thèse, nous présentons notre travail pour améliorer l'efficacité des tests exploratoires du système.

Définition 2.1.2 (Cas de test)**.** Un ensemble de conditions préalables, de données d'entrée, d'actions(le cas échéant), de résultats attendus et de postconditions, élaboré sur la base des conditions de test.

Définition 2.1.3 (Suite de test)**.** Ensemble de cas de test ou de procédures de test à exécuter dans un cycle de test spécifique.

Pour tester un logiciel, il est préférable d'exécuter une suite de cas de test. Distribuer les objectifs sur un ensemble de cas de test permet d'isoler les défauts.

Définition 2.1.4 (Oracle de test)**.** Une ressource pour déterminer le résultat attendu à comparer avec le résultat observé du système. Il peut s'agit d'une autre version du système, un individu ou un ensemble de référence, mais pas du code.

Définition 2.1.5 (Couverture de test)**.** Le degré, exprimé en pourcentage, selon lequel un élément de couverture spécifié a été exécuté lors d'une suite de test, par exemple, l'ensemble des branches d'un code source.

2.1.2 Test Exploratoire

Le Test Exploratoire (TE) est introduit par Cem Kaner [32] sans définition formelle. L'idée d'exploration dans les pratiques de test des logiciels est alors de donner la liberté au testeur de concevoir et exécuter ses tests. Avec l'idée que ces deux activités s'enrichissent l'une l'autre. La notion d'apprentissage du système et de la façon de le tester est primordiale, et se poursuit tout au long du projet.

James Bach [7] en donne une courte définition : "Le test exploratoire est un apprentissage, une conception de test et une exécution de test simultané". L'objectif du TE est de découvrir des bugs par l'exploration, plutôt que par une couverture systématique du système. Il permet aussi aux testeurs de développer leur compréhension du système. Cette connaissance peut servir à proposer des améliorations ou à réaliser des tests plus pertinents. Le TE peut être vu comme une approche complémentaire des techniques systématiques utilisant des cas de tests scriptés issus des spécifications.

Les praticiens des tests scriptés signalent que la conception de cas de test est un problème difficile et que l'avantage de construire des suites de test avec un niveau de détail élevé est perçu comme faible [31]. Dans une discussion sur les besoins des entreprises en matière de pratiques de test, Antonia Bertolino décrit un écart entre la recherche sur les tests de logiciels et les pratiques industrielles. La recherche a activement développé des techniques basées sur la couverture de test depuis les années 1990. Cependant, une grande partie de l'industrie ne mesure pas la couverture de test autre que la couverture de branche sur le code source. Elle souligne le fait que, d'une part, les praticiens disposent d'un temps très limité pour les tests, et que, d'autre part, les approches de test proposées ont un impact sur l'ensemble du cycle de vie du logiciel en exigeant une documentation et une modélisation étendues.

Les résultats expérimentaux d'Afzal et Itkonen [4, 30] montrent que l'efficacité des tests exploratoires est comparable à celle des tests scriptés. Avec l'avantage de soulever moins de faux positifs, c'est-à-dire des tests qui échouent alors que le logiciel fonctionne correctement. En outre, les bugs découverts lors des sessions de tests exploratoires sont plus complexes à déclencher, nécessitant des séquences d'interactions plus longues pour les découvrir. Pfahl et al. ont mené une étude sur l'utilisation du TE dans l'industrie [48]. Leurs résultats indiquent qu'il est utilisé pour tous les types de tests et devrait être davantage outillé.

Bach introduit le concept de session de TE qui agrège les tests d'une équipe de testeur. Le déroulement d'une session de TE est défini dans une charte de test [7].

L'efficacité des sessions de TE dépend fortement de l'expérience des testeurs [8]. Un testeur expérimenté peut ainsi produire des tests plus efficaces lorsqu'il en a la liberté, de l'autre côté, un testeur débutant aura besoin d'être guidé [45, 8, 24]. Ghazi et al. proposent de classer les types de charte de test en fonction de la liberté laissée aux testeurs, allant du test libre à une approche complétement scriptée.

Définition 2.1.6 (Charte de test)**.** Une charte de test est un document écrit par le manager de test. Elle définit l'objectif et le cadre d'une session de TE. L'objectif peut être de

tester une certaine fonctionnalité, chercher un type de problème ou vérifier un ensemble de corrections de bug.

La charte de test cadre la session de test exploratoire. Le contenu d'une charte de test peut varier d'une entreprise à l'autre. Ghazi et al. ont identifié 35 éléments qui peuvent être utilisés dans les chartes de test [23], par exemple la durée de la session, une partie du système sur lequel se concentrer, la liste des bugs trouvés précédemment, des notes issues de sessions de test précédentes, une description des principaux scénarii d'utilisation. Les éléments de la charte de test peuvent être utilisés pour accompagner des testeurs durant la session de test, en fixant un cadre plus ou moins restrictif sur la façon de réaliser les tests. Ghazi et al. présentent un compromis entre la liberté donnée aux testeurs et leur expérience avec le test exploratoire [24]. Plus un testeur est expérimenté, plus il bénéficie d'un niveau de liberté élevé, à l'inverse un testeur connaissant mal l'application sera plus efficace si des indications précises lui sont données sur la façon de chercher des bugs. Dans le cas le plus extrême, la session de test est entièrement scriptée et les testeurs exécutent pas à pas les cas de test. À l'inverse, les sessions *freestyle*, sans restrictions, permettent aux testeurs expérimentés de s'appuyer sur leurs connaissances pour trouver plus de bugs.

2.2 Applications Web et Comportement Utilisateur

Pour utiliser une application web, l'utilisateur utilise un client qui lui-même interagit avec un ou plusieurs services web, via des Application Programming Interface (API). Le travail de cette thèse se concentre sur le test en boîte noire de ces applications. Les tests sont effectués à partir de l'interface graphique, via un navigateur web, soit par un humain, soit par programmation en utilisant un web driver[1] Les tests en boîte noire permettent de tester sans avoir à prendre en compte la répartition entre les différents services web ou les différentes technologies qu'ils peuvent utiliser. Dans cette section,

1. https://www.w3.org/TR/2013/WD-webdriver-20130117/

nous définissons les termes utilisés pour décrire les interactions du testeur avec l'application, ainsi que les outils qui existent pour enregistrer et exécuter les interactions avec les applications web.

2.2.1 Terminologie

Définition 2.2.1 (Application Web)**.** Une application web est un ensemble de programmes sur un ou plusieurs serveurs distants, avec lesquels l'utilisateur interagit via un navigateur web.

Une application web n'a pas besoin d'être installée sur la machine de l'utilisateur puisqu'elle s'exécute sur un serveur distant. L'utilisateur accède à l'application via un navigateur web qui affiche l'interface sous la forme d'une page web.

Définition 2.2.2 (page web)**.** Une page web est une interface graphique affichée par le navigateur de l'utilisateur, elle est composée d'un document HyperText Markup Language (HTML), de fichiers JavaScript utilisés pour modifier dynamiquement le document HTML et interagir avec le serveur, et de Cascading Style Sheets (CSS) utilisées pour la mise en forme du document.

Nous utilisons les suites événements produites par l'utilisateur lors de sa navigation sur une page web pour décrire son comportement sur l'application.

Définition 2.2.3 (Evenement)**.** Un événement e est déclenché par la page web pour indiquer qu'une action a été réalisée sur la page. Il peut s'agir par exemple d'un clic sur un bouton, d'un texte tapé dans un champ de formulaire. Les événements peuvent être captés par du code JavaScript et déclencher des traitements asynchrones susceptibles de modifier l'état de la page web, ou d'envoyer des requêtes Hypertext Transfert Protocol (HTTP) au serveur. Nous considérons qu'un événement est caractérisé au minimum par un type (ex : clic, touche du clavier enfoncée) et une cible (ex : bouton, zone de texte).

Les événements sont habituellement utilisés pour modifier l'état de la page web en réaction aux actions de l'utilisateur. La navigation de l'utilisateur est donc générale-

ment représentée par des événements dans les spécifications par une machine a état fini [9, 41, 58]. L'utilisation de ces modèles pour la génération de séquences d'événements maximisant la détection de bugs a été largement étudiée. Des chercheurs ont développé des techniques utilisant des machines à état [17, 18], des grammaires [55, 40], des algorithmes génétiques [34], des modèles probabilistes [57]. Ces techniques permettent de générer des cas de tests pour différents domaines. Les travaux de cette thèse se différencient, car nous utilisons des modèles pour assister les testeurs, plutôt que de générer entièrement les suites de test.

2.2.2 Enregistrer les actions utilisateur

Le suivi des actions des utilisateurs peut être appliqué avec différents objectifs. Par exemple, le test d'utilisabilité cherche à identifier les améliorations dans l'utilisation d'un produit logiciel ou à identifier les schémas d'utilisation fréquents. L'enregistrement des traces de l'utilisateur est une technique souvent utilisée pour le test automatisé des applications web [19]. Un outil de capture enregistre les actions des utilisateurs (clics de souris, touches du clavier, commandes de navigation, etc.) pendant que l'application est en cours d'utilisation. Les séquences d'actions peuvent ensuite être rejouées par le navigateur. L'intérêt porté à la capture et la rejouabilité des séquences d'action est mise en avant par le nombre d'outils existant dans la recherche et dans l'industrie, parmi eux Mugshot, [46], Timelapse [15], CoScripter [37], Jalangi [52], Selenium [2].

Le suivi est effectué en injectant du code JavaScript dans les pages Web, qui est chargé d'enregistrer les événements reçus par la page Web. L'injection de code JavaScript dans la page web peut se faire en utilisant des proxies HTTP, qui interceptent la page HTML et ajoutent un en-tête JavaScript avant de la transmettre au client [6], ou en utilisant une extension sur les navigateurs web [15]. Atterer et al. [6] définissent les exigences suivantes pour une approche non invasive du suivi des actions sur une application web.

— Un enregistrement détaillé des actions utilisateurs sur la page, les mouvements de

souris ou le scrolling, ainsi que les entrées réalisées sur le navigateur, comme les clics ou les entrées clavier.

— Une indépendance vis-à-vis de la technologie utilisée par le serveur et la partie client.

— Aussi peu de modifications de la partie client que possible.

— Une transparence de l'exécution, l'expérience d'utilisation de l'application ne doit pas être perturbée.

— Pas de changement de la partie serveur. Une méthode de tracking doit pouvoir s'appliquer sur une application dont le serveur n'appartient pas à la personne enregistrant les actions.

2.3 Modélisation de l'utilisation du système

L'utilisation des modèles probabilistes construits sur des séquences d'événements et appliqués aux tests des logiciels n'est pas nouvelle [57, 56, 13, 59, 59]. Nos travaux reprennent ces outils, pour modéliser le comportement des utilisateurs du système. Dans cette section, nous introduction les notations utilisées pour décrire les processus stochastiques, puis présentons leur application au test des logiciels.

2.3.1 Terminologie

La modélisation stochastique est un terme générique dont le but principal est de représenter un comportement par des modèles probabilistes.

Définition 2.3.1 (Espace d'états)**.** L'espace d'états S est l'ensemble de tous les résultats possibles d'une expérience.

Définition 2.3.2 (Événement stochastique)**.** Un événement x est un sous-ensemble de l'espace d'états. $P(x)$ est la probabilité que x fasse partie de l'ensemble des résultats de l'expérience.

Dans l'exemple d'un lancer d'une pièce, l'espace d'échantillonnage est $S = pile, face$, Il peut être séparé en deux événements $A = pile$ et $B = face$. Si la pièce est équilibrée, $P(a) = P(b) = 0.5$. Appliqué aux logiciels, seules des valeurs numériques peuvent être générées, par exemple une distribution uniforme des réels sur l'intervalle $[0, 1]$. Lorsqu'un utilisateur ouvre la page d'authentification d'une application web, l'ensemble des actions disponible est $seconnecter, s'inscrire$. Dans le cas d'un nouvel utilisateur, on pourra dire que $P(s'inscrire) = 1$ Dans nos travaux, nous utilisons la notion de variable pour représenter le caractère incertain des actions d'un utilisateur.

Définition 2.3.3 (Variable aléatoire)**.** Une variable aléatoire X est une fonction $X : S \to \mathrm{R}$ qui associe à chaque événement un nombre réel.

Nous utilisons la notion de variable aléatoire pour définir les chaines de Markov.

2.3.2 Chaines de Markov

Définition 2.3.4 (Modèle de Markov (MM) du premier ordre, ou Chaine de Markov)**.** Une chaine de Markov à valeur dans S est un processus stochastique à temps discret noté $(X_n)_{n\in\mathbb{N}}$ tel que :

— pour tout entier $n \in \mathbb{N}, X_n$ est une variable aléatoire dans S.

— pour tout $n \in \mathbb{N}$, pour tout $(n+1)$-uplet à valeur dans S, $(x_0, x_1, ..., x_n)$, vérifiant $P(X_0 = x_0, ..., X_{n-1} = x_{n-1}) \neq 0$, on a
$P(X_0 = x_0, ..., X_{n-1} = x_{n-1}, X_n = x_n) =$
$P(X_n = x_n | X_{n-1} = x_{n-1}) ... P(X_1 = x_1 | X_0 = x_0) P(X_0 = x_0)$

On représente souvent les chaînes de Markov à l'aide d'un graphe sur l'espace d'état, appelé graphe de transition. La Figure 2.1 montre une chaine de Markov sur l'alphabet $\mathrm{E} = A, B, C, D$.

— Les sommets du graphe sont tous les états $x \in S$.

— Les arêtes orientées sont les couples (x, y).

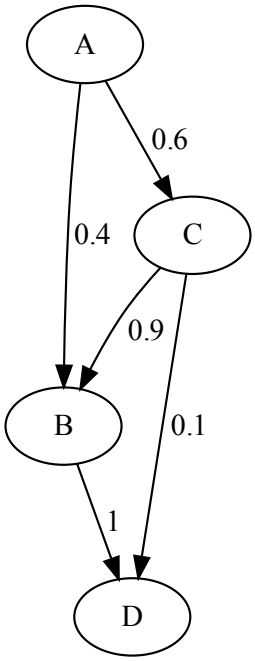


FIGURE 2.1 – Une chaine de Markov sur l'alphabet $\mathrm{E} = A, B, C, D$

— Sur les arêtes orientées, on indique la probabilité de transition correspondante $P(X_{n+1} = y|X_n = x)$. Si la probabilité vaut 0, l'arête n'est pas marquée.

Les chaines de Markov du premier ordre expriment que le futur dépend uniquement de l'état actuel, et pas des états précédents. D'un point de vue pratique, cela signifie que seul l'état courant a besoin d'être connu pour prédire l'état suivant.

Un outil pour enregistrer les probabilités et les utiliser est la matrice de transition.

Définition 2.3.5 (Matrice de transition)**.** Soit l'alphabet $\mathrm{E} = x_1, ..., xn$, La matrice de transition $|\mathrm{E}| \times |\mathrm{E}|$ se défini comme

$P_{i,j} = P(X_{n+1} = x^j|X_n = x^i)$

2.3.3 Modèles de Markov d'ordre supérieur

Les MM de premier ordre permettent de construire un modèle simple. Dans notre cas les applications que nous modélisons peuvent être complexes et inclure plusieurs pages web, chacune pouvant recevoir différents événements. Pour représenter le cheminement

d'un utilisateur dans une application, nous considérons qu'il faut prendre en compte plus que l'état actuel.

Définition 2.3.6 (Model de Markov d'ordre k)**.** Un modèle de markov d'ordre k est un processus stochastique qui suit la propriété suivante : $P(X_{x+1} = x_n + 1|X_n, ..., X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n, ..., X_{n-k+1} = x_{n-k+1})$ pour tout $n \in \mathbb{N}$

Notation 1. Soit un MM d'ordre N, Nous notons P_{MM^N} la probabilité calculée à partir d'un MM d'ordre N.

L'ordre N d'un MM défini la taille de la mémoire et la complexité. La complexité est bornée par $mn + 1$ avec m la taille de l'alphabet.

Définition 2.3.7 (N-gram)**.** On appelle N-gram une séquence de N événements consécutifs. Dans notre application, il s'agit de séquence d'interactions avec le système.

L'utilisation d'une mémoire longue permet de modéliser avec plus de précision, mais fait aussi perdre au modèle sa capacité à donner une prédiction pour une séquence d'événements qu'il a déjà rencontrés, mais de taille inférieure à sa mémoire. Par exemple, si un MM d'ordre 2 a enregistré les séquences d'actions $< e_1, e_2, e_3 >$ et $< e_1, e_4, e_2 >$, il ne pourra pas donner de prédiction pour une séquence finissant par $e_3, e_2 >$, là où une mémoire de taille 1 donne un résultat non vide.

2.3.4 n-gram interpolé

Lorsqu'un modèle du système est inféré à partir de traces d'utilisation, celui-ci est une estimation et par nature incomplet. Si une valeur faible est choisie pour n, les statistiques obtenues seront fiables, car issues d'un nombre élevé d'occurrences, à l'inverse, lorsque le contexte sur lequel se base la prédiction est court, il arrive que le contexte permettant de réaliser la séquence d'action ne soit pas capturé, alors il est probable de

générer des séquences d'actions qui ne peuvent pas s'exécuter sur le système. Augmenter la taille de N permet de capturer un contexte plus grand, et de s'assurer que les dépendances entre les événements soient respectées. Cependant, cela diminue le nombre d'occurrences de chaque séquence, lorsque comparé au nombre total de N-grams. Tonella et al. [54] propose l'utilisation de n-gram interpolés, qui prennent avantage des N-tuples de longue taille lorsqu'ils sont disponibles, puis utilise les valeurs de N plus petites lorsqu'une valeur supérieure n'est pas possible. L'idée est d'interpoler les probabilités pour le prochain événement e, avec k allant de 1 à $N-1$, $P(e|e_1, e2..., e_k)$, avec $k \in [1, ..., N-1]$. La probabilité interpolée est calculée à partir des probabilités conditionnelles pour les contextes de tailles allant de 1 à $N-1$ multipliées par un poids $w(k)$, qui augmente avec la valeur de k, dans notre cas de façon exponentielle. De cette façon les probabilités s'appuyant sur des N-grams de longue taille ont une importance significativement plus élevée. La probabilité conditionnelle pondérée W_k se calcule suivant la formule

$$W_k(e|e_1, e_2, ..., e_k) = 2^k P(e|e_1, e_2, ..., e_k) \tag{2.1}$$

Les probabilités conditionnelles sont ensuite additionnées et normalisées, résultant en une probabilité conditionnelle P^* pour le prochain événement e, étant donné la séquence d'événement précédent $(e_1, e_2, ..., e_{N-1})$

$$S^*(e|e_1, e_2, ..., e_{N-1}) = \sum_{k=1}^{N-1} W_k(e|e_1, e_2, ..., e_k) \tag{2.2}$$

$$P^*(e|e_1, e_2, ..., e_{N-1}) = \alpha S^*(e|e_1, e_2, ..., e_{N-1}) \tag{2.3}$$

Avec α un facteur de normalisation calculé en faisant la somme des valeurs S^* pour les événements pouvant suivre la séquence de taille N-1.

$$\alpha = 1/ \sum_{e \in succEvents(e_1, e_2, ..., e_{N-1})} S^*(e|e_1, e_2, ..., e_{N-1}) \tag{2.4}$$

Dans nos travaux, nous utiliserons la notion de N-grams interpolés pour décrire le

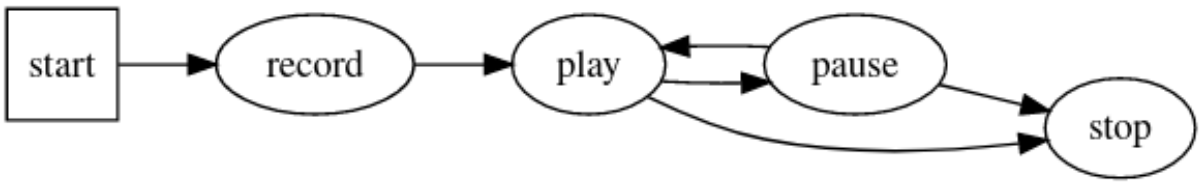


FIGURE 2.2 – Un graphe de flux d'événements

comportement d'un utilisateur sur le système.

2.3.5 Application des modèles pour le test d'interface web

Memon et al. proposent en 2001 l'utilisation d'un graphe de flux d'événement pour modéliser la structure de l'interface graphique, mesurer la couverture de test pour les tests d'interface graphique [42]. Par exemple la Figure 2.2 représente le graphe de flux d'événement pour un système de lecture de piste. Des critères peuvent ensuite être appliqués pour couvrir les sommets ou les arrêtés du graphe de flux d'événements.

Définition 2.3.8 (Graphe de Flux d'Événement)**.** Un graphe de flux d'événement d'un composant d'interface graphique est un couple $< V, E >$ avec

— V, l'ensemble des sommets représentant les événements de A.

— $E \subseteq V \times V$ est l'ensemble des arêtes orientées entre les sommets. On dit qu'un événement e_j suit un événement e_i si et seulement si e_j peut être exécuté immédiatement après e_i. Une arête $(v_x, v_y) \in E$ si et seulement si v_y suit l'événement v_x.

AutoBlackTest [39] et GUITAR [26] sont des outils permettant d'automatiquement générer un modèle d'une interface graphique. GUITAR est composé d'un Ripper, chargé de faire une rétro-ingénierie de l'interface. En partant de la première page de l'application, il interagit avec les composants de l'interface pour construire un graphe d'intégration contenant les relations entre les pages de l'application. GUITAR construit également un Graphe de flux d'événements pour l'application. AutoBlackTest modélise l'application

en utilisant des méthodes de Q-learning en découvrant comment naviguer dans l'application et gérer des situations complexes comme remplir des formulaires, sauvegarder des fichiers. Le modèle inféré est un graphe orienté où chaque sommet est un état de l'interface, et les arêtes sont les événements permettant de changer d'état, associés à un score d'utilité appelé Q-value entre 0 et 1. La valeur associée à une action dépend de l'impact que l'action a sur l'état, par exemple cliquer sur un bouton pour ajouter un événement au calendrier affiche le calendrier entier, ce qui représente un grand changement dans l'interface et est perçu comme très intéressant pour le test. À l'inverse, cliquer sur une combo-box ne provoque peu de changement et aura une valeur faible. La génération des tests est faite par un Q-Learning Agent, qui parcours le modèle pour un nombre paramètre d'épisodes, la longueur de chaque épisode est aussi un paramètre.

Brooks et Memon présentent l'utilisation d'un profil d'utilisation pour construire un MM appelé un graphe probabiliste de flux d'événements [13]. Les séquences d'événements réalisées par les utilisateurs forment un profil d'utilisation et sont découpées en n-gram pour construire la matrice de transition du MM.

$$r_1 = INIT, e_1, e_2, FINAL$$
$$r_2 = INIT, e_1, e_3, e_1, e_2, FINAL$$
$$r_3 = INIT, e_1, e_2, e_2, e_2, FINAL$$

Soit E l'ensemble des événements réalisés par l'utilisateur. Soit $count(e_i)$, la fonction retournant le nombre d'occurrences de e_i dans l'ensemble des séquences d'événements réalisés par les utilisateurs. La probabilité qu'un événement choisi au hasard dans une séquence utilisateur est :

$$\frac{P(e_i) = count(e_i)}{\sum_{j=1}^{E} count(e_j)}$$

Soit un profile d'utilisation contenant les 3 séquences d'actions $< r_1, r_2, r_3 >$, on a :

$$count(e_1) = 4; P(e_1) = 0.4$$
$$count(e_2) = 5; P(e_2) = 0.5$$
$$count(e_3) = 1; P(e_3) = 0.1$$
$$count(INIT) = 3; P(e_3) = 0.1$$
$$count(FINAL) = 3; P(e_3) = 0.1$$

Le précédent calcul est maintenant est étendu aux séquences d'événements de taille S. En prenant $S = 2$, on obtient les séquences :

$$s_1 = e_3, e_1$$
$$s_2 = e_1, e_2$$
$$s_3 = e_2, e_2$$
$$s_4 = e_{INIT}, e_1$$
$$s_5 = e_2, e_{FINAL}$$

L'exemple suivant applique count() et le calcul des probabilités en considérant des séquences de taille 2 au lieu des événements seul.

$$count(s_1) = 1; P(s_1) = 0.11$$
$$count(s_2) = 3; P(s_2) = 0.33$$
$$count(s_3) = 2; P(s_3) = 0.22$$
$$count(s_4) = 3; P(s_3) = 0.33$$
$$count(s_5) = 3; P(s_3) = 0.33$$

Brooks et Memon utilisent ce modèle pour générer des cas de test en concaténant les séquences d'événements de taille S ayant la plus forte probabilité de se produire. Dans notre travail, nous reprenons la construction d'un MM à partir d'un profil d'utilisation et la combinons avec l'utilisation de N-gram interpolés.

2.4 Synthèse

Le test des applications via leur interface graphique est un sujet complexe de par le nombre de combinaisons d'interactions possible, et la difficulté de construire et maintenir un modèle de l'application. La construction d'un modèle pour générer des tests et mesurer la couverture du système par une suite de test demande un travail de modélisation qui peut être difficile à mettre en place pour une entreprise. La pratique du test exploratoire permet de tester le système sans s'appuyer sur des cas de test spécifiés au préalable. C'est une pratique efficace qui permet de raidement obtenir un résultat. Cependant l'efficacité du test exploratoire est fortement dépendante de la compétence des testeurs. Nous pensons que l'outillage des sessions de TE permet de réduire l'impact personnel du testeur, et ainsi permettre une efficacité plus homogène de la pratique. Les modélisations basées sur les chaines de Markov permettent de construire un modèle à partir des séquences d'actions ayant été exécutées par des utilisateurs ou des testeurs. Nous proposons de construire un modèle en enregistrant les séquences d'actions réalisées par les testeurs pour construire pendant les sessions de test exploratoire. Puis d'utiliser ce modèle pour assister les testeurs et ainsi améliorer l'efficacité des sessions de test exploratoire.

Chapitre 3

AIFEX

Sommaire

Dans ce chapitre, nous décrivons le fonctionnement général du socle logiciel développé durant cette thèse pour mettre en application nos approches d'assistances aux tests exploratoires. AIFEX (Artificial Intelligence for Exploratory Testing) assiste les testeurs directement dans leur navigateur lors de sessions de tests exploratoires d'applications web.

3.1 Vue d'ensemble

AIFEX est un outil opensource construit pour servir de socle d'expérimentation. Son développement a débuté au cours de cette thèse et continuera d'évoluer dans à travers de futurs travaux. AIFEX se découpe en deux composants principaux : le serveur web et l'extension de navigateur [1].

Le rôle principal d'AIFEX est d'enregistrer des sessions de test exploratoire, puis d'utiliser ces sessions pour fournir des recommandations aux testeurs directement dans l'interface. AIFEX est composé de trois parties principales : Le *listener*, le *highlighter* et les *services web* chargés de fournir des recommandations. Le listener est chargé de suivre les interactions effectuées par le testeur et de les convertir en une séquence de mots qui sera utilisée dans les services web. Le listerner écoute les événements DOM et capte donc cet événement de clic. Il utilise un ensemble de règles de mappage pour transformer cet événement DOM technique en mot abstrait 'AddToBasket'. Enfin, il ajoute ce nouveau mot à ceux qui ont été générés depuis le début du test et envoie le message aux services web. Les *Services Web* gèrent l'enregistrement des séquences d'actions et les systèmes de recommandations pour la session de TE. Ils prennent en entrée une séquence de mots et produisent les prédictions pour les prochains mots qui peuvent apparaître. Le *highlighter* met en évidence les éléments DOM qui présentent un intérêt pour la session de TE et précise davantage leur recommandation. Il utilise l'ensemble des règles de mappage de l'application web pour identifier les éléments DOM correspondant à chaque mot abstrait recommandé. Ensuite, il applique une surcouche graphique pour indiquer à l'utilisateur les interactions recommandées.

La figure 3.1 montre les différents composants d'AIFEX et leurs interactions. Le service Session est chargé de l'enregistrement des explorations, sous la forme de séquences de mots, réalisés par les testeurs. Le composant Diversité correspond au système de recommandations présenté dans le Chapitre 4. Le composant Scénarios correspond au composant présenté dans le Chapitre 5.

1. Le code source de AIFEX est disponible à l'adresse https://github.com/labri-progress/AIFEX

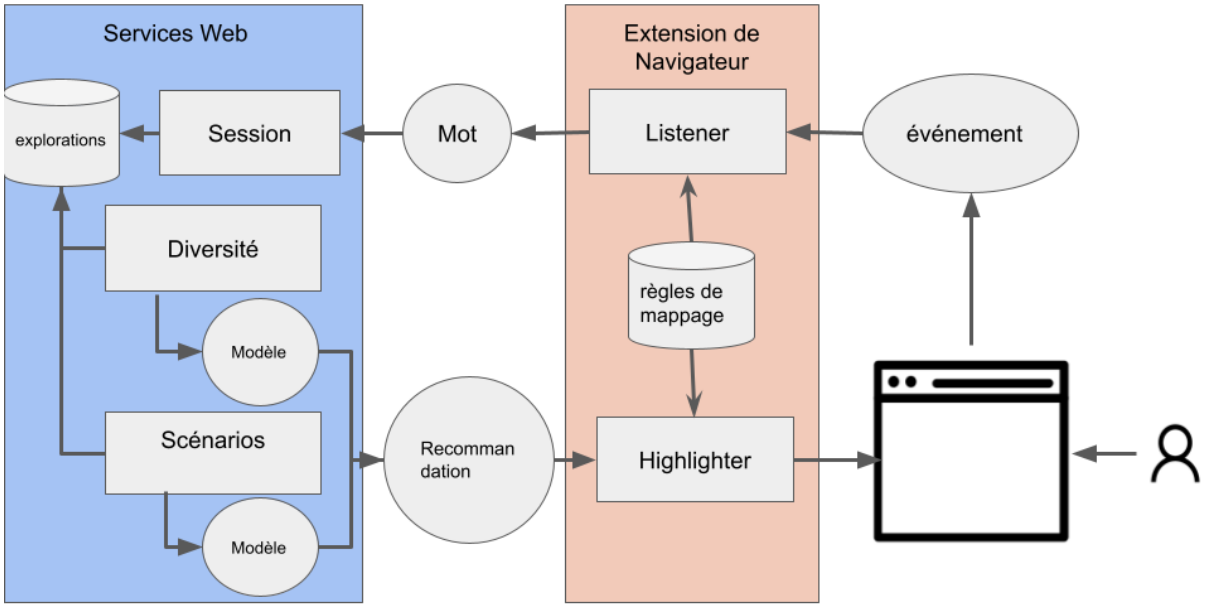


FIGURE 3.1 – Schéma des interactions entre les composants d'AIFEX.

3.2 Services Web

L'objectif de AIFEX est de fournir aux testeurs des informations et recommandations directement dans l'interface web sur les interactions qu'ils peuvent réaliser. Pour cela, AIFEX utilise un ensemble de services web développé durant cette thèse. Chaque service s'exécute dans un conteneur Docker et propose une API. Dans cette section, nous décrivons les deux principaux service servant de base à notre approche, *site web* et *session*, qui sont chargés de la configuration des sessions de tester et le stockage des tests exploratoire réalisés durant les sessions de test.

3.2.1 Site Web

Le service *Site Web* permet de définir comment AIFEX doit reconnaitre et enregistrer les actions réalisées par les testeurs sur un site web. Le module utilise des **règles de mappage**, permettant d'un côté d'identifier les événements dans la page web, et de l'autre abstraire cette action pour la rendre interprétable. Une action est un mot clé

décrivant l'intention métier derrière l'interaction réalisée par le testeur. Par exemple, un clic sur le bouton panier d'une plateforme de commerce en ligne peut recevoir le mot clé $ClickBoutonPanier$.

Les **règles de mappage** s'écrivent dans un format JSON, et se décomposent en trois sections, le *match*, l' *output* et le *context* (optionnel). La partie *match* indique comment reconnaitre les événements JavaScript devant être capturés par la règle de mappage. La partie *output* décrit comment convertir l'événement JavaScript en un mot clé. La partie *context* agit comme une garde sur l'application de la règle et permet de filtrer les règles appliquées sur la page web, il permet de réduire le surcoût dû à l'utilisation d'AIFEX en réduisant le nombre de règles de mappage à évaluer.

Listing 3.1 – *Action - Recherche d'un produit* mapping rule

```
{
    "context": {
        "url": "www.cdiscount.com"
    },
    "match": {
        "event": "keydown",
        "code": "Enter",
        "selector": ".hSrcInput > input"
    },
    "output": {
        "prefix": "ChercherProduit",
        "suffix": "value"
    }
}
```

— *context* est optionnel. Il peut être défini par une *URL*, ou un sélecteur xpath ou css. Les sélecteurs sont des expressions permettant d'identifier un ensemble de nœuds dans le DOM de la page web. Lorsqu'un contexte est défini, cela signifie que le

reste de la règle de mappage ne s'applique que si l'URL courante est incluse dans celle du contexte. S'il s'agit d'un sélecteur, le reste de la règle ne sera évalué que si l'ensemble des nœuds du DOM identifié par ce sélecteur contient au moins un élément. Dans l'exemple de la règle de mappage 3.1, la règle ne s'applique que sur les pages de CDiscount.

— Un *match* défini les conditions que l'événement JavaScript doit remplir pour que l'action doit reconnue. Lorsque l'événement est détecté sur un élément de la page identifié par le sélecteur, alors un mot clé est produit.
  — Un *event* correspond au type d'événement [2] attendu.
  — Un sélecteur *css ou xpath*. Pour qu'un événement déclenche la règle de mappage, il faut qu'il cible un élément capturé par le sélecteur. Les éléments sont obtenus en utilisant la fonction JavaScript querySelectorAll()
— Un *output* définit le mot qui va être construit à partir de l'événement JavaScript. Le mot est toujours composé d'un préfixe, avec un suffixe optionnel. Lorsqu'un suffixe est défini, il est séparé du préfixe dans le mot produit par le caractère "$".
  — Le *prefix* est une chaine de caractère obligatoire.
  — Le *suffix* est un mot clé optionnel, qui permet d'ajouter un paramètre au mot produit. La valeur du suffixe est construite à partir de l'élément du DOM identifié par le sélecteur du *match*. Dans l'exemple du listing 3.1, le suffixe sera construit à partir de l'ensemble d'éléments du DOM identifié par le sélecteur ".hSrcInput > input"
    — innerText : Le mot produit aura comme suffixe le texte contenu dans l'élément DOM cible par l'événement capturé.
    — index : Le mot produit aura comme suffixe l'indice de l'élément DOM visé par l'événement dans la liste des éléments retournés par le match. Si l'appelle à querySelectorAll retourne 3 éléments, un clic sur le deuxième produira la valeur 2.

2. https://www.w3.org/TR/uievents/

— value : Le mot produit aura comme suffixe la valeur de l'attribue "value" du premier élément de la liste retournée. Dans l'exemple du listing 3.1, cela correspond au texte tapé dans la barre de recherche.

3.2.2 Session

Le service *Session* est chargé de l'enregistrement des interactions des testeurs avec l'application durant une session de test exploratoire. Le service reçoit les séquences d'actions depuis l'extension de navigateur décrite dans la Section 3.3. Les séquences d'actions sont alors appelées **explorations**, et peuvent contenir des actions obtenues après l'application des règles de mappages décrites dans la Section 3.2 ou de commentaires, laissés par les testeurs.

Des commentaires accompagnés de captures d'écran peuvent être enregistrés par le testeur via l'extension de navigateur. Ils sont alors associés à une interaction dans l'exploration et peuvent être replacés dans le contexte des actions ayant précédé l'écriture du commentaire. De cette façon, nous pouvons indiquer à un testeur réalisant une exploration qu'un autre testeur ayant réalisé une séquence d'action similaire a laissé une observation sous la forme d'un commentaire. Il peut alors confirmer ou non l'observation.

3.3 Extension de Navigateur

L'extension de navigateur de AIFEX s'installe sur les navigateurs des testeurs pour exécuter un programme en tâche de fond, appelé Background. Le Background est chargé d'injecter un script dans les pages web. Il contient deux composants : Le Listener, et le Highlighter. Le Background sert ensuite d'interface entre les services web d'AIFEX décrits dans la Section 3.2 avec le Listener et le Highlighter. Le Listener est chargé d'appliquer les règles de mappage pour abstraire les actions réalisées par les testeurs, à partir des événements JavaScript qui traversent la page web. Le Highlighter, est chargé

d'afficher une surcouche visuelle à la page web pour guider les testeurs. L'identification des éléments pointés par le Highlighter peuvent être utilisés pour augmenter la diversité des actions réalisées au cours de la session, ou pour aider le testeur à réaliser l'objectif fixé pour la session. Le choix des actions à pointer au testeur est décrit dans les chapitres 4 et 5.

3.3.1 Listener

Le Listener est, avec le Highlighter, un des deux composants s'exécutant dans la page web. Son rôle est d'écouter les évènements traversant le DOM, et de trouver les règles de mappages dont le match capture l'événement. Il produit alors le mot en utilisant la partie output de la règle et l'envoi au Background. Une des difficultés techniques est que les événements peuvent déclencher un changement de page web, et ainsi détruire le contexte, et l'exécution du script. Comme c'est par exemple le cas lorsque l'on clique sur un lien vers une autre page. Il est alors important que l'enregistrement de l'action soit fait avant le changement de page. Pour capter les événements, le Listener attache un EventListener sur l'élément racine du DOM.

3.3.2 Highlighter

Le Highlighter est le deuxième composant exécuté dans la page Web. Il est chargé d'ajouter la surcouche graphique pour guider les testeurs. La figure 3.2 montre une page du site web de CDiscount monitoré par AIFEX. Certains éléments de la page ont un cadre bleu, indiquant que ces actions sont utilisées par une règle de mappage. En plus de cette superposition graphique, d'autres indications peuvent être ajoutées. Sur la figure, certains éléments ont un cadre rouge ou vert, ce qui correspond à des indications renvoyées par le service web Modèle. La signification de ces couleurs est décrite dans le chapitre 4.

Lors de l'initialisation, le Highlighter injecte une feuille de style dans la page Web, et itère sur les règles de mappage pour ajouter un attribut "$aifex\_style$" et un attribut

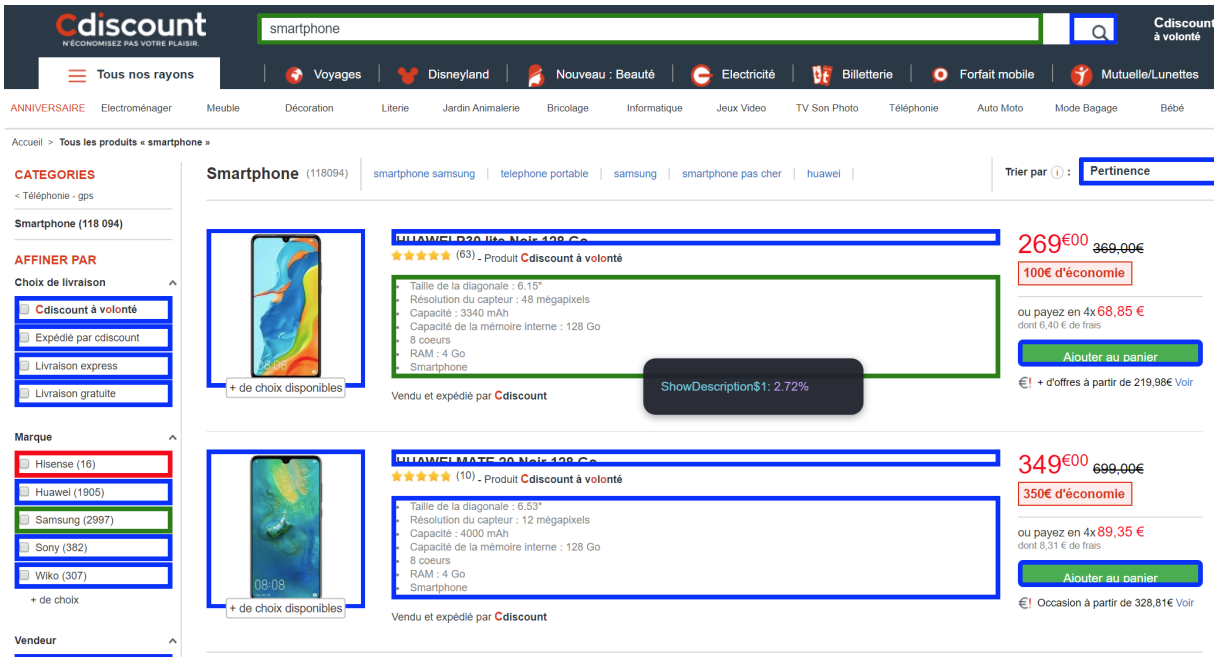


FIGURE 3.2 – Une page web monitorée par AIFEX.

"$aifex_{<}rule\_prefix>$" aux éléments du DOM sélectionnés par la partie match d'une règle de mappage. Pour mettre en évidence les éléments de la page qui correspondent à une action, la règle est identifiée par son préfixe, puis les éléments correspondants dans le DOM sont récupérés via la méthode querySelectorAll("$aifex\_ < rule_prefix >$"). Un style est ensuite appliqué à ces éléments pour leur ajouter un cadre de couleur.

3.4 Évaluation des performances

Notre approche vise à injecter un retour visuel dans l'application web testée. Cependant, ce retour visuel ne doit pas dégrader les performances globales de l'application web pour que notre approche soit utilisable en pratique. Les directives existantes [47] considèrent qu'un délai de moins de 0,1 seconde est considéré comme instantané, tandis qu'un délai de moins d'une seconde est supportable, mais perceptible. Par conséquent, une question importante est de savoir si notre approche peut calculer et injecter le retour visuel en moins d'une seconde. Pour répondre à cette question, nous présentons les

résultats d'une étude approfondie des performances.

3.4.1 Configuration expérimentale

Dans notre mise en œuvre, les principaux composants qui peuvent être affectés par des problèmes de passage a l'échelle sont le Listener et le Highlighter qui vérifient tous deux les correspondances possibles entre les éléments actuels du DOM et les mots abstraits.

En effet, un sélecteur XPath ou CSS doit potentiellement être interrogé sur le DOM actuel pour chaque règle de mappage, ce qui peut être une opération coûteuse. La partie contexte des règles doit réduire le nombre de requêtes à effectuer. Pour étudier les performances de ces composants, nous avons mis en place un grand ensemble contenant 12870 règles de mappage ciblant l'application web Cdiscount.

Les règles de mappage sont séparées en 31 contextes pour identifier les éléments DOM les plus pertinents. Notre scénario est de mesurer le temps nécessaire pour calculer les mappages sur la page d'accueil de l'application web Cdiscount. Nous avons choisi cette page parce qu'elle est l'une des plus compliquées de l'application web, contenant au moment de notre expérience $6,080$ nœuds dans son DOM.

Le temps de calcul est mesuré comme suit. Nous exécutons un script qui installe une version instrumentée de notre extension de navigateur Chromium et rejoint automatiquement une session de test. Lorsque la session est prête, le script pilote Chromium pour ouvrir la page d'accueil de l'application web Cdiscount et récupère le temps de calcul du plugin instrumenté et le nombre de requêtes exécutées sur le DOM. Nous appliquons ce processus 5 fois pour un sous-ensemble de règles croissant de 0 à $12,870$ en ajoutant 100 de règles par tour. Nous présentons la valeur médiane pour chaque lot de 5 mesures sur un sous-ensemble de règles donné. Nous utilisons un ordinateur de bureau avec une configuration standard pour réaliser les expériences (Intel Core i7-7700 @ 3.60GHz avec 8 GB de RAM).

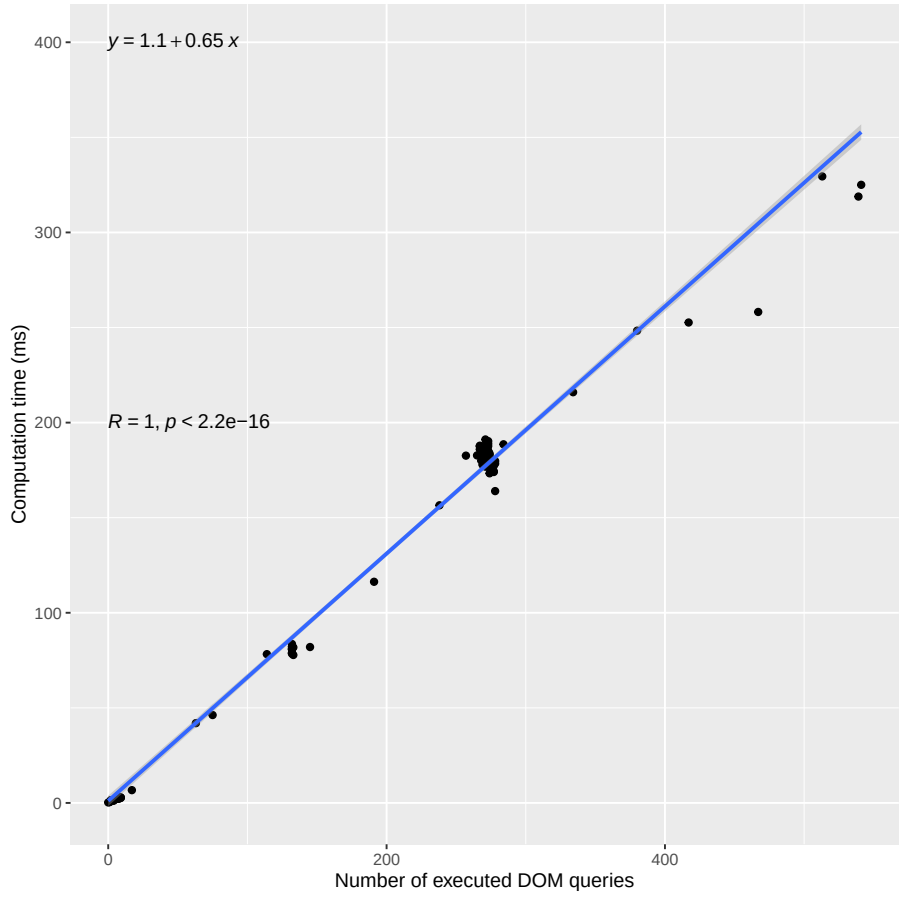


FIGURE 3.3 – Évolution du temps de calcul en fonction du nombre de requêtes DOM.

3.4.2 Résultats empiriques

La figure 3.3 affiche le temps de calcul (axe y) en fonction du nombre de requêtes DOM exécutées (axe x). Comme nous le pensions, le nombre de requêtes DOM exécutées est linéairement corrélé au temps de calcul avec un coefficient de corrélation de Pearson de 1 (R). Comme indiqué dans la formule de régression, le temps de calcul (en ms) est environ 2/3 du nombre de sélecteurs interrogés. Par conséquent, environ 150 sélecteurs peuvent être interrogés en environ $0,1$ seconde et 1500 sélecteurs en environ 1 seconde. Cela signifie que sur l'application web CDiscount, dans le pire des cas où chaque règle de mappage conduit à une requête DOM, nous pouvons traiter au moins 1 500 règles de correspondance.

Une règle de mappage ne sera pas vérifiée si son *contexte* a déjà été interrogé pour une autre règle et que l'on sait qu'il n'est pas présent dans le DOM. Pour évaluer l'effet des *contexte* en pratique, nous affichons dans la Figure 3.4 le temps de calcul (axe y) par rapport au nombre de règles de mappages activées (axe x). Une première observation est que la relation entre le nombre de règles et le temps de calcul n'est pas linéaire puisque nous remarquons une alternance de stagnation et de fortes augmentations. Ce résultat confirme notre hypothèse selon laquelle les contextes permettent de contenir le temps de calcul. Lorsque le *contexte* d'une règle est connu pour ne pas être présent dans le DOM, le temps de calcul n'augmente pas (phases de stagnation). Sinon, lorsque le *contexte* d'une règle n'a pas encore été interrogé, une requête doit être exécutée pour le *contexte* puis une suivante pour le *match* si le *contexte* est présent, ce qui entraîne une surcharge de calcul (phases d'augmentation). Néanmoins, notre implémentation traite l'ensemble complet de 12870 règles en environ $0,3$, seconde, ce qui est bien inférieur à notre délai maximal de 1 seconde. De plus, le seuil de $0,1$ par seconde n'est franchi qu'après le traitement d'environ 4000 règles de mappage. Ces résultats indiquent que notre approche peut s'adapter à l'application web CDiscount, avec un ensemble conséquent de règles (nous rappelons que 12870 règles correspondent à 12870 interactions d'intérêt dans l'application web).

3.5 Synthèse

Nous avons développé un outil permettant de monitor les sessions de tests et afficher des recommandations aux testeurs via une surcouche graphique. Les recommandations sont fournies par des services web indépendants les uns des autres. Cette architecture nous permet d'expérimenter différentes approches pour améliorer l'efficacité des sessions de test exploratoire, et nous pensons que AIFEX a un potentiel réel pour le développement d'approches exploitant les actions réalisées sur une interface web.

AIFEX est construit dans le but de facilement intégrer de nouvelles approches, il nous a servi de plateforme pour nos expérimentations, et nous espérons qu'il sera utilisé pour

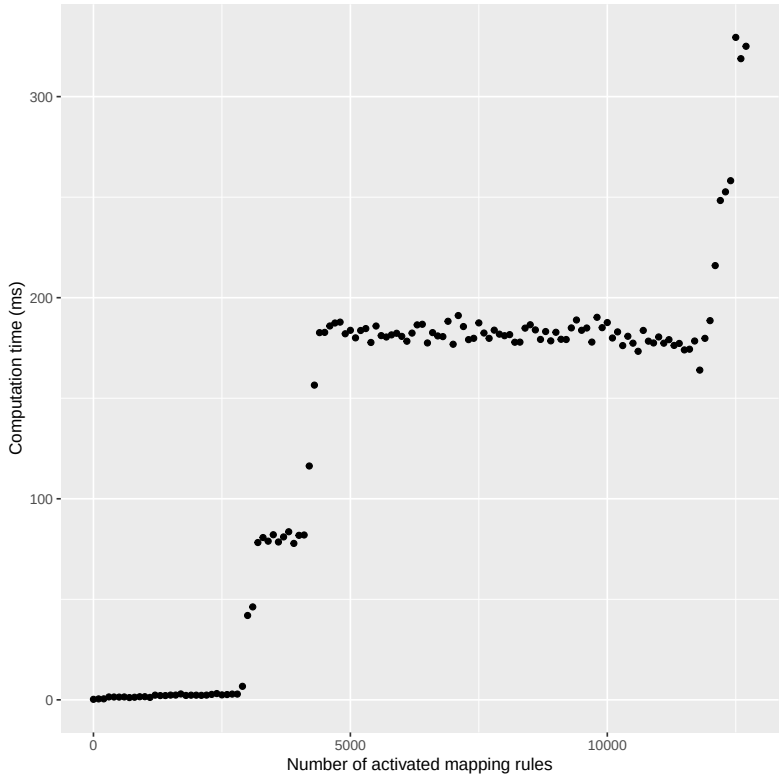


FIGURE 3.4 – Évolution du temps de calcul lors de l'augmentation du nombre de règles de mappage.

de futurs travaux de recherche. Le Chapitre 4 présente un service pour analyser les tests réalisés par les testeurs et fournir une recommandation sur les prochaines actions à réaliser. Le Chapitre 5 présente un service pour suivre la progression des testeurs à travers un scenario de test prédéfini et leur fournir des indications sur les différentes méthodes pour le réaliser. Cette approche est utilisée pour accompagner des sessions de test réalisées via des plateformes de crowdsourcing.

En plus des travaux décrits dans cette thèse, nous avons expérimenté l'utilisation d'AIFEX pour identifier les séquences d'actions les plus souvent réalisées par les utilisateurs d'un site web. Cette information peut ensuite être utilisée pour guider des testeurs. Il est également possible de combiner AIFEX avec l'exécution de tests automatisés pour mesurer la couverture des séquences d'actions les plus fréquentes.

D'autres modules ont été développés pour la génération de tests automatisés et la priorisation de suite de tests, un tableau de bord permet également de configurer les sessions de test. Ces modules ne seront pas décrits dans cette thèse dont le sujet reste le test exploratoire.

Au cours de cette thèse, nous avons développé un outil permettant de surveiller les sessions de test et d'afficher des recommandations aux testeurs via une surcouche graphique. Les recommandations sont fournies par des services web indépendants. Nous expérimentons différentes approches pour améliorer l'efficacité des sessions de tests exploratoires, et nous pensons qu'AIFEX a un réel potentiel pour le développement d'approches qui exploitent les actions réalisées sur une interface web.

AIFEX est construit dans le but d'intégrer facilement de nouvelles approches, il a servi de plateforme pour nos expériences, et nous espérons qu'il sera utilisé pour de futures recherches. Le Chapitre 4 présente un service construisant un modèle à partir des actions des testeurs et de fournir une recommandation pour améliorer la diversité de la session. Le Chapitre 5 présente un service permettant de suivre la progression des testeurs à travers un scénario de test prédéfini et de leur fournir des indications sur les différentes méthodes pour le réaliser. Cette approche est utilisée pour accompagner des sessions de test réalisées via des plateformes de crowdsourcing.

En plus des travaux décrits dans cette thèse, nous avons expérimenté l'utilisation d'AIFEX pour identifier les séquences d'actions les plus souvent réalisées par les utilisateurs d'un site web. Ces informations peuvent ensuite être utilisées pour guider les testeurs. Il est également possible de combiner AIFEX avec l'exécution de tests automatisés pour mesurer la couverture des séquences d'actions les plus fréquentes.

D'autres modules ont été développés pour la génération automatique de tests et la hiérarchisation des suites de tests, ainsi qu'un tableau de bord pour la configuration des sessions de tests. Ces modules ne seront pas décrits dans cette thèse, le sujet restant les tests exploratoires.

Chapitre 4

Améliorer la diversité des sessions de test exploratoire

Sommaire

Dans ce Chapitre, nous décrivons notre contribution pour aider les testeurs à améliorer la diversité de leurs approches de test exploratoire sur les applications web.

4.1 Introduction

Rappelons que le TE est souvent organisé en sessions avec un temps limité [31, 48] dont la durée peut varier de quelques minutes à plusieurs heures. Une session de TE peut impliquer un seul testeur ou un groupe de testeurs en collaboration. Par exemple, une entreprise peut organiser une session d'une journée entière impliquant un groupe de 10 testeurs, tandis qu'une autre entreprise peut demander à ses collaborateurs de réaliser des sessions individuelles quand ils le peuvent. De telles sessions ne durent généralement que quelques minutes.

Au cours d'une session, plusieurs problèmes peuvent affecter l'efficacité des testeurs et, par conséquent, la qualité des tests. Tout d'abord, les testeurs peuvent être biaisés par leurs connaissances et donc passer à côté d'anomalies ou d'interactions avec le système inhabituelles pour eux. Par conséquent, les testeurs peuvent ne jamais effectuer certaines interactions, qui peuvent être à l'origine de certains défauts. Deuxièmement, en raison de la complexité du système, les testeurs peuvent rapidement oublier quels tests ont déjà été effectués. Par conséquent, les mêmes tests peuvent être rejoués plusieurs fois, ce qui fait perdre du temps et réduit l'efficacité des sessions de test. Troisièmement, lorsque les sessions sont réalisées par des groupes de testeurs, ceux-ci peuvent échouer à collaborer et se concentrer sur la même partie système. Nous soutenons que les testeurs impliqués dans une session sont censés favoriser la diversité globale des tests qu'ils effectuent en tant que groupe.

Nous proposons dans ce Chapitre une approche pour aider les testeurs à améliorer la diversité de leurs sessions de test exploratoire sur les applications web. Nous conseillons aux testeurs de réaliser des interactions nouvelles ou rarement réalisées, ce qui augmente la diversité des tests. Pour atteindre cet objectif, nous introduisons un modèle de prédiction basé sur des n-gram interpolés dans la section 2.3.4. Ce modèle est implémenté dans le service web **modèle** d'AIFEX, présenté dans le Chapitre 3. Plus précisément, nous entrainons continuellement un modèle de prédiction en assimilant les traces de test qui ont été précédemment réalisées par les testeurs. Ensuite, lorsqu'un testeur réalise un nouveau scénario de test, nous demandons au modèle de prédiction de proposer les prochaines interactions qui sont celles fréquemment réalisées. Nous conseillons alors au testeur de ne pas suivre la prédiction mais, au contraire, de considérer les interactions ayant un faible pourcentage de prédiction. Si le testeur suit notre conseil, il réalise alors des interactions pas ou peu testées. Une fois que l'interaction est effectuée, nous entraînons à nouveau le modèle de prédiction avec la nouvelle trace de test, qui demandera donc plus de diversité pour les prochains tests.

Le reste de ce Chapitre est organisé comme suit : La Section 4.2 motive notre approche. 4.3 explique comment nous récupérons les actions du testeur et les utilisons pour guider les testeurs. La Section 4.4 évalue l'efficacité de notre approche pour mener des sessions de TEcrowdsourcées. La Section 4.5 évalue l'efficacité de notre approche pour mener des sessions de TEcrowdsourcées. La Section 4.6 Présente une étude de cas en entreprise sur un projet réalisé par CIS Valley. Enfin, la Section 4.7 est une Synthèse de ce travail.

4.2 Motivation

Dans cette section, nous décrivons un exemple de session de TE réalisée sur une application web. Nous utilisons cet exemple pour mettre en évidence les problèmes que nous abordons et pour présenter notre approche.

4.2.1 Exemple de test exploratoire sur une application web de e-commerce

En exemple, nous considérons une session de TE sur le site web de Cdiscount qui est la boutique en ligne de l'un des plus grands vendeurs de commerce électronique en France. La session est réalisée de manière collaborative par Alice, Bob et John, dont le but est d'identifier les anomalies lors de la commande d'un smartphone sur le site web. La session commence avec Alice qui utilise le champ de texte de recherche pour rechercher "smartphone" (cf. Figure 4.1a). Ensuite, après avoir parcouru l'ensemble des produits répertoriés pour évaluer qu'ils sont tous des smartphones (cf. Figure 4.1b – ❶), elle ajoute le premier d'entre eux à son panier ❹. Enfin, dans la dernière étape, après avoir vérifié que le prix est cohérent (cf. Figure 4.1c), elle commande le produit. Pour poursuivre cette session, le groupe de testeurs devra effectuer davantage de tests, ce qui devrait augmenter la diversité et les chances de trouver des défauts. En particulier, Alice, Bob et John devront essayer plusieurs filtres ❷, visualiser et sélectionner parmi différents produits listés ❸ et ❹, et choisir différentes options ❺. La réalisation de tests aussi diversifiés peut cependant devenir problématique pour les trois raisons que nous avons brièvement présentées dans la section 5.

Tout d'abord, un testeur peut être biaisé par sa connaissance du ST et peut donc manquer ou ignorer certaines interactions. Dans notre étude de cas, une requête de recherche peut être effectuée en remplissant le champ de texte de recherche puis appuyer sur la touche "entrée", ce que font la plupart des utilisateurs. Cependant, une autre possibilité consiste à remplir le champ de texte de recherche puis à cliquer sur le bouton de la loupe qui se trouve à côté du champ de recherche. Cette interaction peut facilement être ignorée par les testeurs, surtout s'ils ont l'habitude d'appuyer sur la touche "Entrée". Deuxièmement, la complexité de l'application peut rendre difficile, voire impossible, pour les testeurs de se souvenir des tests qu'ils ont déjà effectués au cours de la session. Dans notre scénario, il existe de nombreux filtres permettant d'affiner les résultats de recherche de produit. Les filtres proposés dépendent des mots clés

tapés dans la barre de recherche, et il est presque impossible de savoir lesquels ont été sélectionnés ou non, ce qui empêche le testeur de tester efficacement toutes les possibilités. Enfin, lorsque plusieurs testeurs veulent collaborer dans une session, ils doivent savoir quels sont les scénarios que les autres testeurs ont explorés ou sont en train de faire. Lorsqu'une telle connaissance ne peut être obtenue, le système peut être décomposé en plusieurs sous-parties et chaque testeur est affecté à une sous-partie spécifique. Par exemple, Alice peut être affectée aux filtres ❷, Bob à la sélection du produit ❸ et John aux options ❺. Cependant, une telle décomposition est difficile à réaliser en pratique et empêche de couvrir les tests de bout en bout tests à cheval sur la décomposition adoptée.

4.2.2 Exploiter les scénarios testés pour recommander des interactions non couvertes.

Pour favoriser la diversité du TE, nous soutenons que les testeurs devraient être encouragés à privilégier les interactions qui n'ont pas (ou rarement) été réalisées plus tôt dans la session.

Ces conseils peuvent être facilement donnés si un modèle du système existe et si une métrique de couverture de test peut être calculée. Si tel est le cas, la couverture de test peut rendre compte des composants et des interactions du système qui ont été testés. Conseiller les testeurs consisterait alors à les inviter à effectuer les parties et interactions non couvertes.

Cependant, le modèle de l'application et la couverture de test sont rarement disponibles dans le contexte du TE. De plus, la construction automatique d'un modèle d'une application web existante est un problème très complexe, qui n'est pas encore résolu [44]. Les applications web, en particulier les *single page applications* (SPA) [43], utilisent fortement les événements asynchrones, dont les effets modifient l'interface utilisateur graphique de manière non déterministe, ce qui rend ce problème encore plus difficile. Au début, notre exemple courant semble être composé de seulement trois pages

web, avec des interactions clairement identifiées.

Cependant, la deuxième page (cf. Figure 4.1b) est directement liée à la requête de recherche et change constamment lorsque l'utilisateur met à jour le filtre. Comme elle ne peut pas être identifiée comme une page unique, la génération automatique d'un modèle à partir d'elle aboutirait à un nombre illimité de pages similaires, mais différentes. Par conséquent, nous supposons que *i)* le modèle du système n'est pas disponible, et *ii)* la couverture de test ne peut être calculée.

Notre proposition s'appuie sur le *Traitement du langage naturel* et, plus précisément sur les n-gram, mais déforme leur utilisation pour favoriser la diversité. Plus précisément, nous considérons que les tests qui ont été précédemment réalisés peuvent être intégrés dans un modèle de Markov pour prédire les interactions futures. Cependant, un tel modèle de prédiction favorise les interactions qui ont été fréquemment réalisées dans les tests précédents. Par exemple, si l'on considère qu'Alice a entraîné le modèle de prédiction avec son premier scénario, le modèle ne pourra alors proposer que des interactions réalisées par Alice (puisqu'il ne connaît que celles-ci). Nous proposons alors de nous éloigner de cette prédiction et d'inviter le testeur à considérer des interactions avec une probabilité faible ou même sans prédiction. Par exemple, si Bob commence le deuxième test de la session, le modèle de prédiction lui conseillerait de commencer par rechercher 'smartphone', car c'est ce qu'a fait Alice. En analysant toutes les interactions que Bob pourrait effectuer, nous pouvons l'inviter à cliquer plutôt sur le bouton de la loupe, qui a une probabilité nulle en tant que score de prédiction. En effet, en agissant ainsi, il peut contribuer à augmenter la diversité de la session.

4.3 Améliorer la diversité des sessions de test exploratoire

Notre modèle de prédiction se base sur les modèles de langage n-gram interpolés présentés dans la Section 2.3.4, qui sont une classe de modèles probabi-

listes qui supposent que l'on peut prédire la probabilité de certains mots futurs en regardant seulement quelques mots dans le passé [14]. Avec un gram n (de taille n), si s est une séquence de mots ($s = w_1, \ldots, w_{m-1}$, avec $m > n$), alors $P(w_m|s) = P(w_m|w_{m-n}, \ldots, w_{m-1})$ fait référence à la probabilité d'observer l'ajout de w_m à la séquence connue s. Dans notre contexte, un mot clé w abstrait un événement JavaScript capté par le Listener de AIFEX décrit en section 3.3. Une séquence s est un scénario de test en cours d'exécution. Un modèle de langage à n-grams peut alors prédire ce que devrait être la prochaine action en considérant simplement les $n - 1$ actions passées. Les prédictions sont ensuite utilisées par le Highlighter de AIFEX (voir Section 3.3 pour identifier quels sont les événements JavaScript correspondants et, par conséquent, quels éléments du DOM peuvent être ciblés pour les déclencher.

L'une des limites à l'utilisation des modèles de langage à base de n-gram est que la valeur de n doit être déterminée *a priori*. Dans notre contexte, si n est petit, alors peu d'interactions passées sont nécessaires pour prédire ce que devrait être la prochaine action. Avec notre exemple, si $n = 2$ par exemple, alors une seule interaction antérieure est nécessaire, ce qui n'est pas assez important pour savoir où se trouve le testeur dans le tunnel de commandes du produit. Si n est trop grand, par exemple $n = 10$, alors 9 interactions précédentes sont nécessaires et devraient être les mêmes que celles effectuées par le testeur pour prédire la prochaine. Une telle condition se produirait rarement et limiterait donc la fiabilité des prédictions. Pour remédier à cette limitation, nous utilisons des n-gram interpolés, permettant de choisir un intervalle plutôt qu'un seul n, et d'utiliser les prédictions de chaque k de l'intervalle.

Alice	SearchProduct$smartphone, AddToBasket$1, ShowBasket, Order
Bob	SearchProduct$smartphone, AddToBasket$1, ShowBasket, RemoveItem$1
John	SearchProduct$smartphone, ShowDescription$1, ShowBasket, Order

FIGURE 4.2 – Trois tests réalisés par Alice, Bob et John

La figure 4.2 illustre trois scénarios de test réalisés par Alice, Bob et John au début de leur session de TE. Nous considérons que ces trois scénarios de test ont été enregistrés dans un modèle unique partagé pour la session de test en cours. Maintenant, nous considérons qu'Alice lance un nouveau test et qu'elle commence par déclencher l'interaction 'SearchProduct$smartphone'. Si nous demandons au modèle une prédiction, il recherchera tous les k-grams dont les $k-1^{nieme}$ mots correspondent au test effectué par Alice. Comme Alice n'a effectué qu'une seule interaction à partir de maintenant, le modèle recherchera tous les 2-grams dont le premier mot correspond à 'SearchProduct$smartphone'. Trois 2-grams seront retournés : ⟨SearchProduct$smartphone, AddToBasket$1⟩ deux fois, et ⟨SearchProduct$smartphone, ShowDescription$1⟩. La prédiction pour les prochaines interactions sera donc : 'AddToBasket$1' avec $0,66$ de probabilité, et 'ShowDescription$1' avec $0,33$.

On considère maintenant qu'Alice continue sa session, qui devient : SearchProduct$smartphone, AddToBasket$1 et ShowBasket. Si nous demandons à nouveau au modèle de faire une prédiction, il cherchera des 2-grams, des 3-grams et des 4-grams, car Alice a maintenant effectué 3 interactions. Il renverra deux 4-grams (⟨SearchProduct$smartphone, AddToBasket$1, ShowBasket, Order ⟩), qui correspondent au test précédemment effectué par Alice et Bob, et un 2-grams issu du test de Bob : ⟨ShowBasket, Order⟩. La prédiction favorisera les 4-grams en multipliant leur probabilité par 2^4 (la probabilité des 2-grams est multipliée par 2^2). La prédiction donnera donc comme résultat 'Order' avec environ

$0,55$, et 'RemoveItem$1' avec $0,45$.

Le système de prédiction, comme l'illustre cet exemple, a tendance à inviter les testeurs à effectuer les actions qui ont été fréquemment réalisées lors des tests précédents. Cependant, pour favoriser la diversité, nous invitons plutôt le testeur à effectuer des interactions à faible probabilité, et même des interactions qui ne sont pas encore prédites, mais qui peuvent encore être effectuées. Dans cette optique, après chaque interaction, le Highlighter (cf. Section 3.3.2) utilise la partie match des règles de mappage pour identifier tous les éléments qui peuvent être la cible d'un événement JavaScript d'intérêt dans la page actuelle (cf. Section 3.2.1). Ensuite, grâce à la partie output des règles, nous pouvons générer des recommandations d'actions à réaliser par le testeur.

Par exemple, avec Alice, lorsque le modèle de prédiction renvoie 'Order' et 'RemoveItem$1' avec une probabilité de probabilité $0,55$ et $0,45$, elle pourrait également effectuer l'interaction 'QuantitySelect', qui sert à modifier la quantité des produits sélectionnés. Cette interaction est identifiée grâce aux règles de mappage définies pour ce site web, mais n'a pas été prédite par le modèle et n'a donc pas de probabilité. Notre système indique donc cette interaction à Alice en plus des autres.

4.4 Étude Contrôlée (Étudiants)

Ce Chapitre présente 3 validations pour notre approche. La Section 4.4 décrit une étude contrôlée réalisée avec des étudiants de Master 2 spécialité Génie logiciel de l'université de Bordeaux. La Section 4.5 est une seconde étude contrôlée réalisée avec des testeurs recrutés via une plateforme de crowdtesting. La Section 4.6 présente une étude de cas réalisée avec notre partenaire industriel CIS Valley.

Dans cette section, nous décrivons une étude contrôlée que nous avons réalisée pour étudier l'hypothèse selon laquelle notre approche peut mieux guider un groupe de testeurs pour explorer des applications Web avec plus de diversité.

4.4.1 Configuration des expérimentations

Pour ce cela, nous avons recruté 39 étudiants sur les 42 inscrits en dernière année d'un cursus de génie logiciel. Nous avons réparti aléatoirement ces étudiants en 13 groupes de 3 étudiants. Nous voulons étudier l'impact de notre approche sur la diversité des scénarios de test explorés par un groupe de testeurs au cours d'une session. Pour ce faire, nous avons mis en place un plan à mesures répétées [51]. Ainsi, chaque groupe effectue une session avec notre assistant et une session sans.

Pour mettre en place des sessions de tests exploratoires réalistes, nous avons utilisé une application web en production : Cdiscount, une application de commerce électronique. Les règles de mappage utilisées sont décrites en Annexe A . I . Comme objectif de test, nous avons demandé aux groupes d'étudier le pipeline de recherche et de commande de Cdiscount, tel que décrit précédemment dans la Figure 4.1. Nous avons présenté à chaque groupe les éléments de l'interface graphique qui faisaient partie de la session de test, puis nous leur avons demandé de trouver les problèmes d'utilisabilité de cette partie de l'application. Chaque groupe a effectué 2 sessions de 10 minutes : une avec notre assistant, une sans. Notre assistant a utilisé un modèle de prédiction configuré avec une longue de n-gram maximale de 8, et nous avons écrit les règles de mappage pour l'application. L'ordre dans lequel les groupes ont effectué les sessions assistées et non assistées a été décidé de la manière suivante. Dans le programme du cours, il y avait 2 classes à 2 jours d'intervalle. Par conséquent, nous avons arbitrairement choisi une classe pour effectuer la session assistée en premier et l'autre classe pour faire l'inverse. Enfin, parmi les 13 groupes d'étudiants, 5 groupes ont commencé à utiliser l'assistant et 8 groupes ont commencé sans. Pour chaque groupe, nous enregistrons les séquences de test produites lors des deux sessions.

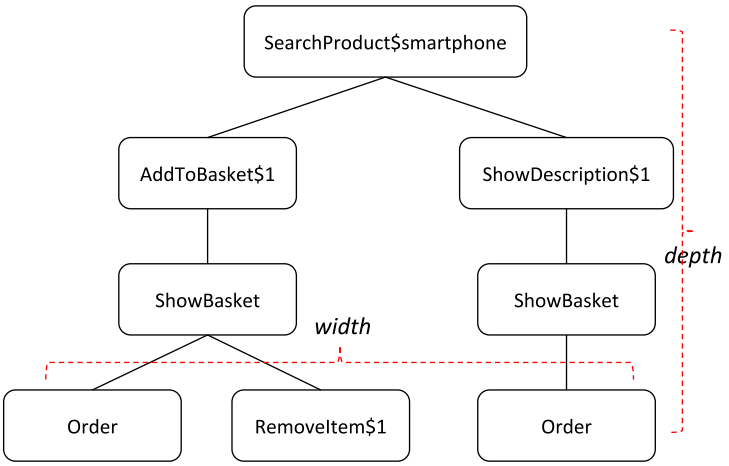


FIGURE 4.3 – Arbre de navigation de la session de la Figure 4.2.

Notre objectif est de mesurer l'impact sur la diversité des tests réalisés lors d'une session de test donnée. Pour définir une métrique quantifiable à cet effet, nous introduisons le concept d'*arbre de navigation*, qui est calculé à partir des séquences de test réalisées au cours d'une session. Cet arbre contient les préfixes de toutes les séquences de test exécutées pendant la session, comme le montre la figure 4.3. À partir de l'arbre de navigation, nous dérivons 2 métriques. Premièrement, la *largeur d'une session* (S_w), qui est définie comme la largeur maximale de l'arbre de navigation. Deuxièmement, la *depth of a session* (S_d), qui est définie comme la profondeur maximale de l'arbre de navigation. Les deux valeurs S_w et S_d sont des entiers positifs. Leurs valeurs dans notre exemple d'arbre de navigation sont illustrées dans la Figure 4.3. Plus la largeur et la profondeur sont grandes, plus les tests effectués au cours de la session sont diversifiés. En plus de ces deux métriques, nous en introduisons une troisième qui vise à quantifier la quantité de répétitions effectuées par un groupe de testeurs au cours d'une session. Cette mesure, appelée *ratio de diversité* ou S_u, analyse tous les n-gram calculés pendant une session et est définie comme le nombre de n-gram distincts divisé par le nombre de tous les n-gram. S_u est un nombre réel compris entre 0 et 1, 1 signifiant que toutes les explorations sont uniques (comme tous les n-gram sont uniques) et une valeur proche de zéro indique que de nombreux n-gram ont été extraits de nombreuses fois, indiquant ainsi une forte répétition. Par conséquent, plus le ratio de diversité est élevé, plus la

diversité des tests effectués au cours d'une session est grande.

Ensuite, nous définissons 3 hypothèses nulles et alternatives :

— H_0^1 : L'utilisation de notre assistant n'a aucun effet sur la largeur d'une session et H_a^1 l'utilisation de notre assistant augmente la largeur des sessions,

— H_0^2 : L'utilisation de notre assistant n'a aucun effet sur la profondeur d'une session et H_a^2 l'utilisation de notre assistant augmente la profondeur des sessions,

— H_0^3 : L'utilisation de notre assistant n'a aucun effet sur le rapport de diversité d'une session et H_a^3 l'utilisation de notre assistant augmente le rapport de diversité des sessions.

Pour évaluer nos hypothèses, nous mesurons d'abord pour chaque groupe d'élèves les valeurs S_d^a, S_w^a, et S_u^a (resp. S_d^0, S_w^0, et S_u^0) résultant des explorations qu'ils ont effectuées lors de la session avec l'assistant (resp. sans). Pour évaluer la significativité des résultats, nous utilisons un test de rang signé de Wilcoxon. Ce test est non paramétrique, car nous n'avons pas d'hypothèse sur les distributions de nos mesures. Pour chaque test où nous acceptons l'hypothèse alternative, nous rapportons également la taille de l'effet, en utilisant l'indicateur de Rosenthal r [50] que nous interprétons en utilisant les seuils communs [21] : petit pour $0,1 <= r < 0,3$, modéré pour $0,3 <= r < 0,5$ et grand pour $r >= 0,5$.

4.4.2 Résultats expérimentaux

La figure 4.4 illustre les résultats que nous avons obtenus en ce qui concerne la *largeur*, la *profondeur* et le *rapport de diversité*. Comme tendance générale, nous remarquons que les valeurs pour les sessions assistées des groupes sont très souvent plus élevées que les valeurs pour les sessions non assistées. Pour évaluer la signification de cette observation, nous effectuons trois tests de rang signé de Wilcoxon, comme expliqué précédemment.

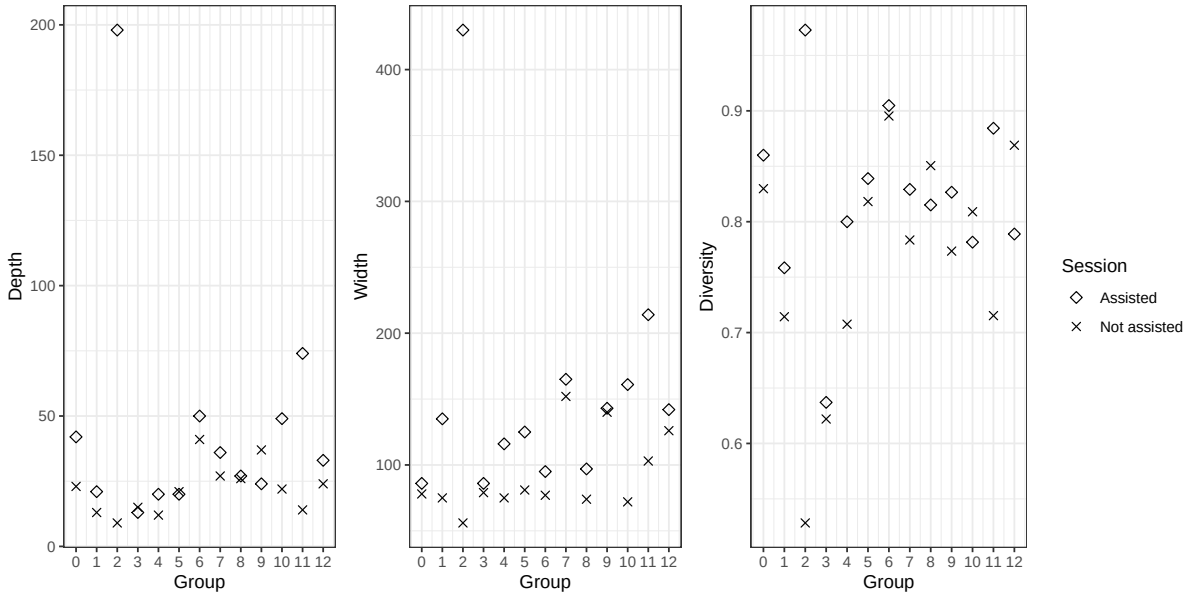


FIGURE 4.4 – La profondeur, largeur, et diversité des sessions assistées et non-assistées des 13 groupes.

Concernant la *profondeur*, la p-value du test est de 0.01371, nous rejetons donc l'hypothèse nulle et acceptons l'hypothèse alternative. La taille de l'effet est de $r = 0.621$ que nous pouvons interpréter comme grande. Concernant la *largeur*, la p-value du test est de 0.0001221, donc nous rejetons l'hypothèse nulle et acceptons l'hypothèse alternative. La taille de l'effet est de $r = 0.882$ que nous pouvons interpréter comme grande. Concernant le *ratio de diversité*, la p-value du test est de 0.04016, donc nous rejetons l'hypothèse nulle et acceptons l'hypothèse alternative. La taille de l'effet est $r = 0.494$ que nous pouvons interpréter comme modérée.

En conclusion, notre assistant a un effet *significatif et important sur la profondeur et la largeur* de l'arbre de navigation extrait d'une session. Les sessions assistées conduisent à des arbres de navigation avec une largeur et une profondeur plus grandes. Notre assistant a également un effet *significatif et moyen sur le ratio de diversité* calculé à partir d'une session. Les sessions assistées conduisent donc à des ratios de diversité plus importants. Tous ces résultats corroborent notre hypothèse selon laquelle notre assistant aide les groupes de testeurs à réaliser des tests plus diversifiés.

4.4.3 Menaces sur la validité

En ce qui concerne la validité interne, nous avions 5 groupes qui ont commencé l'expérience en utilisant notre assistant, et 8 sans. Si nous supposons que notre conception expérimentale peut souffrir d'un effet d'épuisement ou d'apprentissage, cela pourrait avoir influé sur les résultats. Cependant, la différence entre notre situation et la situation la plus équitable (6 vs 7 groupes) est faible, donc l'effet de ces biais est faible à notre avis. En ce qui concerne la validité externe, les participants de l'expérience sont des étudiants qui connaissent bien l'application, mais pas des experts. Par conséquent, rien ne prouve que les résultats seraient généralisés aux testeurs expérimentés ou aux experts du domaine. De plus, notre application Web appartient au domaine de la vente au détail, mais rien ne prouve que nos résultats s'appliqueraient à d'autres types d'applications Web. Enfin, la durée des sessions dans notre expérience était de 10 minutes. Nos résultats pourraient ne pas s'appliquer à des sessions plus longues ou plus courtes.

4.5 Étude Contrôlée (Crowdsourcing)

Dans l'étude contrôlée de la section 4.4, nous avons fait participer des groupes de 3 testeurs à des sessions de 10 minutes de TE de 10 minutes et nous avons mesuré la diversité des explorations produites à la fin de la session. Cependant, dans la pratique, les sessions de TE peuvent impliquer un plus grand nombre de testeurs pendant de plus longues périodes de temps. La capacité de notre approche à augmenter la diversité des tests dans cette configuration reste une question ouverte. Nous procédons donc dans cette section à une deuxième étude contrôlée pour évaluer l'efficacité de notre approche dans une telle configuration.

4.5.1 Configuration Expérimentale

Nous avons conçu le scénario de test sur la base de l'application Web Cdiscount, utilisé un ensemble de 31 règles de mappage similaire à celui utilisé dans l'étude contrôlée

de la section 4.4.

Le scénario du Listing 4.1 comporte plusieurs étapes. Il est demandé au testeur d'augmenter la diversité du scénario entre chaque étape. De plus, certaines étapes peuvent être réalisées de plusieurs façons. Par exemple, Step 4 demande d'ajouter un produit au panier. Sur l'application web, il y a 2 façons d'ajouter un produit au panier, soit directement dans la page de résultat d'une recherche de produit, soit depuis la page du produit. Les testeurs peuvent également utiliser les boutons pour supprimer ou modifier la quantité de produits dans le panier, ou pour revenir en arrière et trouver de nouveaux produits à ajouter dans le panier. Le nombre d'interactions possibles permet de s'assurer qu'il existe plusieurs façons de réaliser le scénario et que les testeurs disposent d'un terrain de jeu suffisamment large pour explorer diverses possibilités.

Listing 4.1 – Le scénario de la session de test exploratoire de l'étude.

```
1  Pour realiser ce scenario, vous devez effectuer les actions
       suivantes dans l'ordre :
2  1. Effectuer une recherche via la barre de recherche en
       tapant un texte dans la barre puis en appuyant sur lea
       touche Entree,
3  2. Ajouter un filtre, ou cliquer sur un menu,
4  3. Aller a la page d'un produit,
5  4. Ajouter un produit au panier,
6  5. Aller au panier,
7  6. Cliquez sur le bouton "Choisir ma livraison".
8 Entre chaque etape, vous pouvez ajouter d'autres actions pour
    enrichir votre scenario.
```

Nous définissons les questions de recherche suivantes :

— RQ_1 : Dans quelle mesure la diversité des tests augmente-t-elle au fur et à mesure de la session ?

— RQ_2 : Notre approche a-t-elle un impact sur la diversité des tests lors de longues

sessions ?

Nous avons créé deux sessions de TE, une assistée par notre approche, et une seconde sans assistance. Nous avons ensuite demandé à une plateforme de crowd sourcing d'effectuer 100 explorations pour chaque session. Nous avons également ajouté la contrainte que les testeurs des deux sessions doivent être différents. Pour les deux sessions, les participants ont reçu le scénario décrit dans le Listing 4.1. En outre, les deux groupes ont été informés qu'ils devaient produire des explorations aussi diverses que possible. Les participants aux deux sessions doivent installer l'extension web d'AIFEX sur leur navigateur (cf. Section 3.3). Dans le cas de la session assistée, le plugin affiche les bordures de couleur pour indiquer les probabilités reçues du modèle de prédiction. Dans le cas de la session non assistée, toutes les bordures sont bleues, et ne montrent aucune information sur la prédiction faite par le modèle de prédiction. Ce comportement garantit que les testeurs non assistés effectueront également des actions qui sont prises en compte par notre configuration. Dans les deux sessions, les instructions demandent d'interagir uniquement avec les éléments identifiés par des bordures colorées, de sorte que toutes les interactions peuvent être mises en correspondance avec des mots. Enfin, nous enregistrons pour chaque session toutes les explorations effectuées par les testeurs, qui sont triées pour chaque groupe en utilisant l'horodatage de la première action effectuée dans l'exploration.

Afin de mesurer l'évolution de la diversité des sessions de test, nous utilisons l'analyse de survie [35]. Pour utiliser l'analyse de survie, nous utilisons cette analogie. Au début d'une session, tous les n-gram réalisables sont "vivants" et le but des testeurs est de "tuer" ces n-gram en effectuant les actions correspondantes dans leurs explorations. Un n-gram donné est considéré comme "tué" dès qu'il a été observé dans une exploration (nous rappelons que les explorations sont triées). Les explorations représentent donc la notion de temps dans notre analyse. En effet, pour augmenter la diversité, le but des testeurs est de tuer le plus de n-gram dans le moins d'explorations possible.

Bien sûr, l'un des problèmes est que l'ensemble des n-gram réalisables est inconnu.

Utiliser l'ensemble de tous les n-gram possibles en utilisant notre alphabet d'actions serait une surestimation grossière. Pour surmonter ce problème, nous utilisons l'ensemble de tous les n-gram observés dans les deux sessions comme ensemble de n-gram réalisables, ce qui est clairement une sous-estimation, mais plus représentatif de la réalité à notre avis. De plus, nous ne pensons pas que l'utilisation de cette méthode pour calculer l'ensemble initial de n-gram ne favorise un groupe dans notre étude.

Enfin, nous extrayons les données suivantes pour notre analyse de survie. Pour chaque n-gram de l'ensemble initial, et pour chaque session, nous enregistrons soit 1) le rang de la première exploration où il a été observé et considérons le n-gram comme "tué" ou 2) le rang de la dernière exploration et considérons le n-gram comme "vivant" (le n-gram sera donc censuré dans l'analyse). Dans notre analogie, un taux de survie plus faible pour les n-gram indique une meilleure diversité pour le test, car plus de n-gram tués signifie que plus de parties de l'application web ont été explorées par les testeurs. Comme nous n'avons aucune hypothèse sur la distribution du taux de survie des n-gram, nous utilisons l'estimateur de Kaplan-Meier non paramétrique commun [33]. Pour évaluer la différence entre le taux de survie de n-gram dans le groupe non assisté et le groupe assisté, nous utilisons le test logrank [11] pour tester l'hypothèse nulle qu'il n'y a pas de différence de taux de survie dans les deux groupes. Bien sûr, la p-value du test est affectée par l'ensemble initial de n-gram qui est sous-estimé dans notre protocole. C'est pourquoi nous indiquons en plus le temps nécessaire pour tuer 50% des n-gram dans les deux groupes comme indication de la taille de l'effet.

4.5.2 Résultats expérimentaux

La figure 4.5 représente la probabilité de survie des n-gram dans les deux groupes en fonction du nombre d'observations. Nous remarquons que pour environ les 20 premières explorations, la probabilité de survie dans les deux groupes est très similaire. Ceci n'est pas surprenant car au début, presque toutes les explorations contiennent de nouveaux n-gram puisque l'ensemble des n-gram observés est presque vide. A partir de la 20^e

exploration, nous observons qu'une différence entre la probabilité de survie dans les deux groupes apparaît, et ce jusqu'à la 40^e exploration. Cela confirme notre hypothèse qu'après un certain temps, les testeurs non assistés commencent à avoir des difficultés à trouver des explorations originales. Entre la 60^e exploration et la 80^e exploration, la différence entre les taux de survie augmente à nouveau en faveur du groupe assisté. Globalement, la différence entre les taux de survie augmente presque toujours en faveur du groupe assisté pendant toute l'expérience.

La valeur p du test de logrank est de $p < 0,0001$, ce qui est largement inférieur au seuil standard de $0,05$, ce qui indique une différence significative entre les groupes. De plus, 50% des n-gram sont tués approximativement en 90 observations pour le groupe assisté alors que, ce nombre n'est jamais atteint pour le groupe non assisté.

Pour en revenir à nos questions de recherche, pour RQ_1 nos résultats indiquent que dans les deux groupes la diversité augmente tant que la session progresse. Les sessions n'ont pas été assez longues pour nous permettre d'observer un plateau de probabilité de survie. En ce qui concerne RQ_2, nous avons observé que notre approche a un effet significatif sur la diversité des explorations, et cet effet semble être de plus en plus fort au fur et à mesure que la session avance. Notre hypothèse est que pour les longues sessions, il devient de plus en plus difficile pour les testeurs non assistés de trouver des explorations originales alors que le retour visuel fourni aux testeurs assistés leur permet de continuer à découvrir des explorations originales.

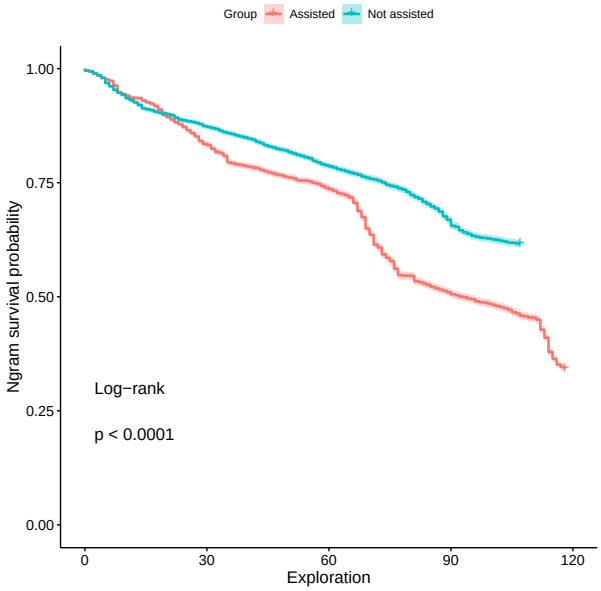


FIGURE 4.5 – Analyse de survie sur les n-gram produits pendant les sessions de test

4.5.3 Menaces sur la validité

Validité interne Les explorations sont classées par horodatage dans notre collecte de données. Cependant, comme le modèle de prédiction n'est mis à jour qu'à la fin d'une exploration, les testeurs assistés effectuant des explorations en meme temps ne tirent pas parti des actions réalisées par les autres testeurs tant qu'ils n'ont pas fini leur exploration. Par conséquent, l'ordre total n'est qu'une approximation basé sur la date d'envoi. Cependant, nous pensons que l'impact est très faible, car cela ne modifie pas la probabilité de survie globale, mais pourrait modifier légèrement le moment où la diminution de la probabilité de survie est observée.

La p-value du test log-rank est affectée par le choix de l'ensemble initial de n-gram comme expliqué précédemment. Si nous avions pris l'ensemble de tous les n-gram possibles, la p-value n'aurait probablement pas été significative (étant donné le nombre de ces n-gram, la probabilité de survie calculée aurait été proche de 100 %. Cette valeur p est donc à prendre avec des pincettes. Afin de laisser les lecteurs décider par eux-mêmes, nous fournissons les courbes de probabilité de survie complètes dans l'article.

Validité externe Dans notre étude, nous faisons appel à des testeurs sous contrat avec une plateforme de crowdsourcing sur une application web de vente au détail. Il s'agit d'un contexte très spécifique, et nos résultats ne peuvent pas être généralisés à d'autres contextes.

4.6 Étude de cas

L'étude de cas dont nous rendons compte dans cet article a été réalisée en collaboration avec CIS Valley, une entreprise ayant une activité de développement de logiciels. CIS Valley cherche à améliorer ses processus de test afin de faciliter l'évolution des produits logiciels qu'elle développe. L'objectif de notre étude de cas était de vérifier que notre approche apporte des bénéfices dans un contexte industriel.

4.6.1 Méthode

La méthodologie que nous avons suivie est composée de trois étapes :

1. Définir les règles de mappage pour SONATE,
2. Exécution de deux sessions de TE (une avec notre approche en mode caché, une avec notre approche en mode affiché),
3. Effectuer une post analyse de la session.

Règles de Mappage.

Afin de délimiter les sessions de TE, et de définir les règles de mappage pour SONATE, nous avons demandé aux managers sur quelle partie de l'application, ils souhaitaient se concentrer. Ils ont choisi la partie utilisée pour créer et modifier les enregistrements correspondant aux individus enregistrés dans l'ERP. D'un point de vue technique, cette partie est composée d'une seule page (Single Page Application), composée de plusieurs composants. Certains composants sont obligatoires et servent à remplir les informations

d'identification de base : prénom, nom, date de naissance, etc. Les autres composants sont facultatifs, et peuvent même être activés ou désactivés dynamiquement.

Heureusement, le backend fournit une API d'introspection qui liste les composants contenus dans une page web donnée de l'application. Lorsqu'elle est appliquée à la page ciblée par notre étude de cas, cette API rapporte tous les composants contenus dans cette page. Nous avons ensuite utilisé cette API pour générer automatiquement nos règles de mappage (cf. Section 3.2.1). De cette manière, nous avons réussi à générer 153 règles de mappage.

Sessions de Test Exploratoire.

Huit employés de l'entreprise ont participé aux deux sessions. Sept d'entre eux étaient des parties prenantes : deux managers, trois consultants d'entreprise et deux membres de l'équipe de migration. L'un d'entre eux n'était plus parti prenante, mais l'était auparavant dans une version précédente du logiciel. Leur niveau de connaissance de SONATE, du point de vue de l'utilisateur, était donc hétérogène. Un testeur a déclaré avoir une très bonne connaissance, un autre a déclaré ne jamais l'avoir utilisé. Les autres testeurs ont une certaine connaissance. Cinq d'entre eux déclarent avoir une bonne connaissance des tests de logiciels et deux des tests exploratoires.

Avant de lancer la première session, nous avons brièvement présenté les principes du test exploratoire. Ensuite, les testeurs ont réalisé une première session de 20 minutes. Pendant cette première session, les testeurs ont utilisé notre approche en mode caché. Ils n'ont pas reçu de prédiction, et ne peuvent pas voir les points forts.

Nous avons ensuite présenté notre assistant avant de réaliser la deuxième session en effectuant une démonstration de 5 minutes sur l'application web Cdiscount. Les testeurs ont ensuite effectué la deuxième session avec notre assistant en mode spectacle. Les testeurs ont reçu des prédictions et ont vu les moments forts. Malheureusement, en raison d'un problème technique, nous avons dû arrêter la deuxième session au bout de 13 minutes, ce qui a un impact sur notre observation, comme nous le verrons plus loin.

Sondage.

Trois semaines après les deux sessions, nous avons soumis une enquête aux testeurs. Nous leur avons demandé leur avis en leur attribuant une note de 1 à 5. Nous avons posé quelques questions sur les tests exploratoires en général, puis plusieurs questions sur notre approche et notre boîte à outils de test. Les questions et les réponses sont disponibles en ligne. [1]

4.6.2 Retours de participants.

Nous avons reçu 6 réponses de la part des 8 testeurs.

D'un point de vue général, il est apparu que l'utilisation des tests exploratoires est une pratique qui les intéresse beaucoup, notamment parce qu'elle permet à des équipes ayant une connaissance hétérogène du ST d'acquérir des connaissances sur l'application cible.

Concernant notre approche, les répondants ont noté $4,5$ sur 5 en moyenne, lorsqu'on leur a demandé s'ils trouvaient utile d'avoir un retour visuel en temps réel des probabilités. Il s'avère que tous ont trouvé utile d'afficher les probabilités avec un code couleur, et que les probabilités étaient considérées comme pertinentes. Ils n'ont pas été perturbés par le fait que les actions doivent être configurées *a priori* via le langage déclaratif que nous proposons. Outre le rendu des probabilités, les répondants ont exprimé le souhait de pouvoir récupérer les séquences d'interactions réalisées lors de la découverte d'un bug, ou encore de pouvoir générer un script de test pour rejouer un scénario de test défaillant.

Les répondants ont également apprécié la nature collaborative de notre outil. Ils ont notamment apprécié le fait que les interactions des autres testeurs soient prises en compte dans les prédictions. Enfin, les répondants ont noté 4 le choix d'un plugin installé directement dans le navigateur, principalement parce qu'ils aimeraient pouvoir effectuer des sessions supplémentaires avec des navigateurs alternatifs.

1. https://researchexperimentation.fr/survey

4.7 Synthèse

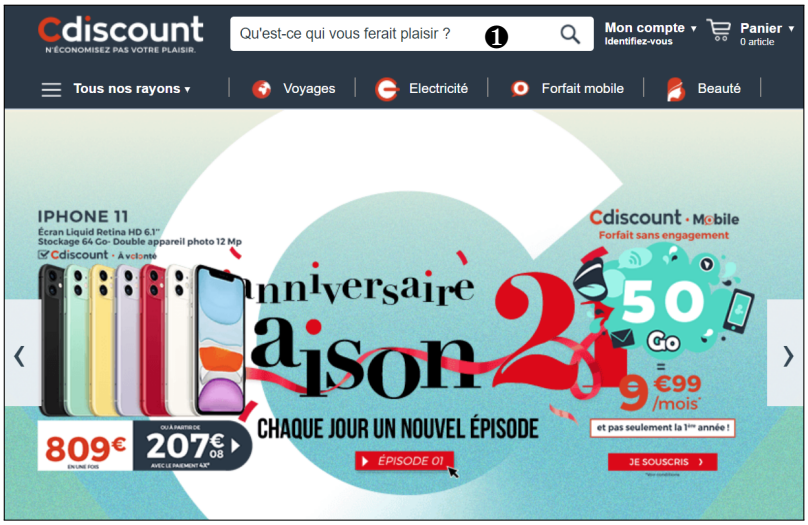
Ce Chapitre présente notre approche pour aider les testeurs qui participent à une session TE à choisir la prochaine la prochaine interaction à réaliser dans le but d'augmenter la diversité de leurs tests. Notre approche est basée sur des modèles de Markov selon lesquels la prédiction d'une interaction future peut être faite en regardant les interactions précédentes. Nous proposons de surveiller les tests effectués par les testeurs, et d'entraîner dynamiquement un modèle. Grâce à ce modèle, nous encourageons les testeurs à effectuer des interactions rarement réalisées pendant la session. Notre approche s'accompagne d'un langage déclaratif utilisé pour définir quelles parties d'un système doivent être surveillées et comment les événements déclenchés par les interactions effectuées doivent être convertis en mots à intégrer dans un modèle n-gram.

Dans le cadre de ce travail, nous avons également développé AIFEX, présenté dans le Chapitre 3, qui supporte entièrement les approches décrites dans cette thèse. Les testeurs sont assistés sont guidés en leurs indiquant quels éléments d'une page devraient être les cibles de futures interactions. Nous avons mesuré les performances de notre extension de navigateur et montré qu'elle est utilisable en pratique pour de grandes applications web, car elle nécessite moins d'une seconde pour afficher une indication.

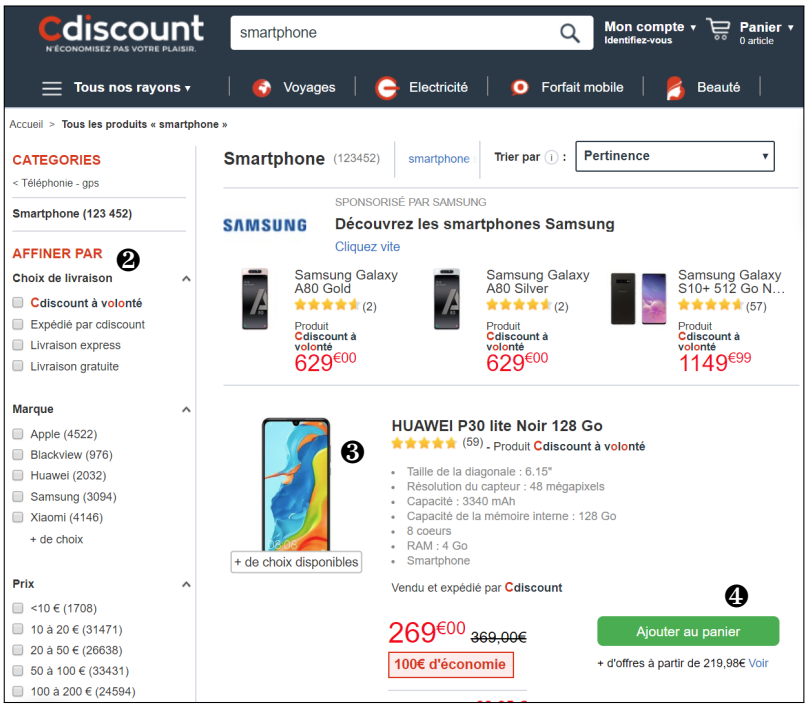
Nous avons ensuite validé notre approche en réalisant deux études contrôlées différentes et une étude de cas. Les études contrôlées ont été réalisées sur une grande application web de commerce électronique. La première montre que notre approche permet aux testeurs d'effectuer des tests plus profonds, plus larges et plus diversifiés tout en participant à de petites sessions (3 testeurs participant à une session de 10 minutes). La deuxième montre que notre approche aide les testeurs à effectuer des tests plus diversifiés lorsqu'ils participent à des sessions TE plus importantes (notre session de crowdsourcing a rassemblé 12 testeurs pendant 8 jours). Cette deuxième étude montre également que plus il y a de tests, plus la diversité est grande. La réalisation de la seconde étude a soulevé des difficultés sur la réalisation d'une session de test exploratoire dans le crowd, et nous aura servi de point de départ pour la motivation du Chapitre 5.

Notre étude de cas a été réalisée sur une PME qui développe actuellement un grand ERP. Elle a montré que notre langage déclaratif peut être utilisé pour exprimer des règles de mappage pour une application web industrielle. De plus, elle a également montré que notre outil peut être utilisé par les industriels qui veulent exécuter des sessions TE.

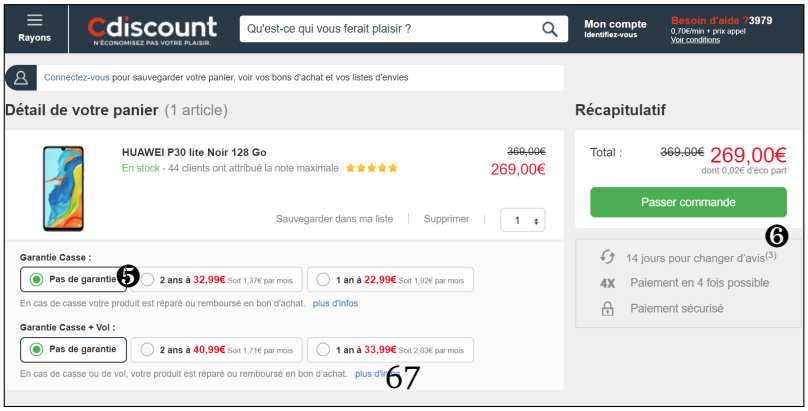
Enfin, notre approche fournit des prédictions, qui peuvent être considérées comme de nouvelles métriques pour les tests effectués dans les sessions TE. Ces métriques invitent les testeurs à changer dynamiquement leur comportement pour favoriser la diversité.



a Recherche d'un produit.



b Filtre, parcours et sélection des produits.



c Validation de la commande.

FIGURE 4.1 – Tunnel de commande de l'application e-commerce de Cdiscount

Chapitre 5

Crowdtesting des sessions de test exploratoire

Sommaire

Le Test Exploratoire repose sur les testeurs : leur expertise et leur capacité d'exploration [31, 8]. Concernant l'expertise du testeur, Ghazi et al. ont montré que les testeurs experts peuvent tirer parti d'un haut niveau de liberté et trouvent plus de défauts lorsqu'ils ne sont pas contraints [24]. En ce qui concerne la capacité d'exploration,

plusieurs travaux ont montré que les testeurs participant à une session ET doivent consacrer du temps à diverses explorations [16, 38]. Lorsque les testeurs experts explorent le système testé de manière extensive, alors l'ET est efficace pour trouver de nouveaux défauts [4, 30, 10].

Cependant, ce qui fait un testeur expert n'est pas trivial. Les testeurs qualifiés qui ont travaillé sur le système pendant plusieurs mois ne sont pas les seuls testeurs experts. Les utilisateurs finaux sont également des testeurs experts, en particulier lorsque la session de TEse concentre sur la fonctionnalité de l'utilisateur. Cependant, organiser une session de TE avec de vrais utilisateurs finaux est difficile et coûteux. L'utilisation d'une plateforme de crowdsourcing peut être une solution alternative si et seulement si le système à tester est une application web grand public, comme Amazon ou Facebook. [22, 25, 20, 36]. Les travailleurs des plateformes de crowdsourcing ont un profil proche des utilisateurs finaux et sont faciles à recruter [28].

Dans ce chapitre, nous proposons une approche pour organiser des sessions de TE à partir d'une plateforme de crowdsourcing. La difficulté d'utiliser une telle plateforme vient de l'expertise des crowdtesters et de leur capacité à explorer l'application web. Les crowdtesters ne sont pas des testeurs experts et des travaux antérieurs ont montré qu'ils ont même des difficultés à suivre les instructions de test [20, 5]. Leur motivation première est de maximiser le profit en accomplissant les tâches le plus rapidement possible, ce qui n'est pas l'état d'esprit du TE. En pratique, lorsqu'on fait appel à des crowdtesters, les chances que les testeurs restent dans le cadre de la session en cours sont faibles. Heureusement, il a été démontré que les testeurs moins expérimentés, comme les crowdtesters, obtiennent de meilleurs résultats lorsqu'ils sont guidés [45]. Notre idée principale est d'introduire une approche qui permet aux concepteurs de sessions exploratoires qui veulent faire appel à des crowdtesters de modéliser formellement l'objectif de la session. Nous tirons ensuite parti de ce modèle pour fournir une orientation visuelle en direct aux crowdtesters qui mènent la session. En utilisant cette approche, nous espérons fortement réduire les chances que les crowdtesters ne suivent pas l'objectif de la session, sans sacrifier la diversité des comportements testés qui sont dans le

cadre de l'objectif de la session.

Notre méthodologie pour valider notre approche est la suivante :

— Nous vérifions d'abord que les crowdtesters ne sont pas efficace lorsqu'on leur demande d'effectuer certaines tâches de TE. Plus précisément, nous répondons à cette question de recherche :

RQ 1 : Les crowdtesters effectuent-ils des explorations correctes ?

— Nous validons ensuite notre approche en répondant à ces questions de recherche :

RQ 2 : Les crowdtesters assistés effectuent-ils plus d'explorations correctes que les non-assistés ?

RQ 3 : Les crowdtesters assistés effectuent-ils des explorations plus diversifiées que les non-assistés ?

.

Nous répondons à nos questions de recherche en validant notre approche sur deux applications web grand public : le site de commerce électronique Amazon et le réseau social Reddit. Nous réalisons ensuite une étude contrôlée impliquant 120 crowdtesters en 4 sessions. Le reste de ce chapitre est organisé comme suit : La section 5.1 motive notre approche. La section 5.2 présente les principes de notre approche, notre extension de navigateur et notre DSL. La section 5.3 présente une étude contrôlée. Enfin, la section 5.4 propose une synthèse de nos travaux.

5.1 Motivation

Dans cette section, nous motivons notre contribution en décrivant une session de TE réalisée sur une application de commerce électronique. La session se déroule sur le site web d'Amazon. Sonia, la responsable de la session de test, définit que la session vise à explorer le tunnel d'achat, à la recherche d'anomalies ou de problèmes d'utilisabilité. Plus précisément, elle définit l'objectif et la portée de la session par le scénario suivant :

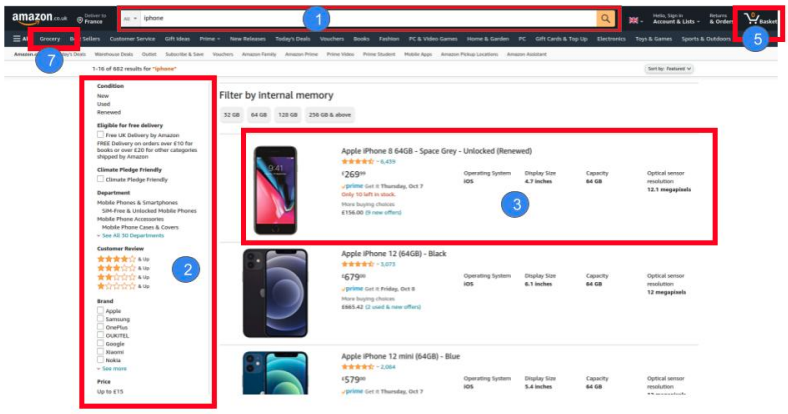
rechercher un produit, utiliser des filtres pour affiner le résultat, ajouter un produit au panier et terminer la commande.

Alice, une experte en tests de l'équipe de Sonia, et Bob, un crowdtester d'Amazon Mechanical Turk, participent à la même session. Ils doivent tous deux effectuer plusieurs explorations, et chacune d'entre elles doit suivre le scénario exprimé par Sonia, mais être différente des précédentes. Le point notable est que Bob reçoit un revenu pour chaque tâche : chaque fois qu'il effectue une exploration.

Alice et Bob lisent tous deux le scénario et commencent leur première exploration. Alice connaît l'application Web, car elle y a travaillé pendant plusieurs mois. Elle commence une exploration en utilisant le mot clé "smartphone" dans la barre de recherche (voir 5.1a–❶). Ensuite, elle sélectionne un ensemble de filtres parmi différentes catégories dans la liste de filtres ❷. En outre, elle vérifie que les résultats sont cohérents lorsqu'un filtre est supprimé et ajouté à nouveau. Ensuite, elle sélectionne un produit en cliquant sur son image ❸. Elle sait qu'en cliquant sur la description ou le titre, elle ouvre également la page du produit. Elle note donc qu'elle utilisera l'un de ces éléments lors de sa prochaine exploration. Elle continue en ajoutant un produit au panier à l'aide de l'un des deux boutons "Ajouter" et ouvre la page du panier à l'aide de l'un des trois boutons. Les autres boutons sont des objectifs à couvrir dans les explorations suivantes. Enfin, elle vérifie qu'elle peut modifier correctement le nombre de produits dans le panier et clique sur le bouton de paiement ❻.

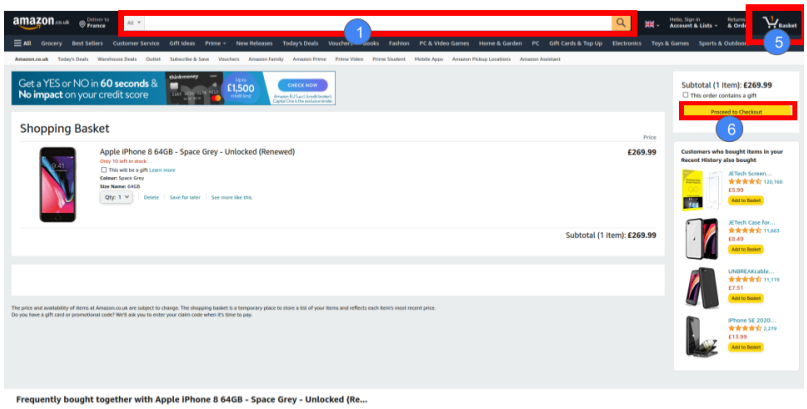
Bob ne sait pas grand-chose de l'application, mais il a peut-être déjà utilisé une application de commerce électronique. Il commence son exploration en tapant dans la barre de recherche. L'étape suivante lui demande d'ajouter un filtre, mais il ne sait pas ce que cela signifie. Malheureusement, il clique sur une catégorie de produits au lieu de cliquer sur ❼. Bob saute alors cette étape du scénario. Il peut donc effectuer une exploration incorrecte selon que cette étape est obligatoire ou non. Sans autre information, il continue à suivre le scénario pour accomplir sa tâche et recevoir le revenu. Finalement, l'exploration effectuée par Bob ne vaut rien du point de vue de Sonia.

Cet exemple est caricatural, mais met en évidence les difficultés d'utilisation des



a Page de résultats de recherche.

b Page-Produit.



c Page panier.

FIGURE 5.1 – Flux de commandes de produits dans l'application e-commerce Amazon

plateformes de crowdsourcing pour le TE. Le point est qu'il y a un compromis à faire entre être encadré par un scénario ou avoir beaucoup de liberté. Il est clair que les crowdtesters doivent être beaucoup plus encadrés que les testeurs classiques, sinon il y a de fortes chances qu'ils fassent des explorations inutiles. Mais ils doivent aussi avoir une certaine liberté pour effectuer des explorations diverses. Enfin, il faut noter que les crowdtesters s'efforcent de terminer la tâche le plus rapidement possible pour obtenir la récompense de la tâche. Pour cette raison, le modèle d'incitation doit prendre en compte la qualité de l'exploration. Cependant, il serait injuste de rejeter la tâche du crowdtester si les instructions ne sont pas claires. Notre proposition, décrite dans la section suivante, est de spécifier le but et la portée de la session à travers un DSL. Cette spécification est utilisée pour guider les crowdtesters à travers les explorations attendues et ainsi recevoir leurs revenus le plus rapidement possible.

5.2 Approche

Dans cette section, nous présentons notre approche pour guider les crowdtesters dans l'exécution du scénario de test. Notre approche consiste à suivre les interactions du testeur avec la page web pour mesurer sa progression. Nous identifions ensuite les éléments de la page qui permettent d'avancer dans le scénario à l'aide d'une animation graphique. Pour ce faire, nous établissons d'abord le scénario souhaité pour la tâche de crowd-testing en utilisant un DSL. Ensuite, nous compilons le scénario dans une machine à états utilisée pendant la session de test pour évaluer la progression dans la tâche de test. Enfin, notre extension de navigateur injecte un feedback visuel directement sur la page web. L'extension affiche une animation pour indiquer les composants permettant de progresser dans le scénario conçu.

5.2.1 Conception du scénario de test.

Le scénario des sessions de test décrit le comportement attendu des testeurs sur l'application. Nous prenons l'exemple de la commande d'un produit sur une application de commerce électronique. Le scénario consiste à rechercher un produit, appliquer des filtres sur les résultats, choisir un produit, et enfin valider la commande. Pour le construire, nous associons d'abord les interactions avec les composants de l'interface graphique en utilisant des événements abstraits. Ensuite, nous utilisons notre DSL pour décrire la structure des séquences d'événements qui forment une exécution correcte du scénario. Ce DSL implémente les opérateurs du langage Auros, utilisé pour le traitement des événements complexes (CEP) [12]. La grammaire du DSL est disponible en Annexe B .. Le scénario est une contrainte exprimée sous forme d'expression algébrique, où les termes sont des événements produits sur la page web. La tâche de crowd-testing est alors validée lorsque le testeur réalise une séquence d'événements qui satisfait la contrainte.

5.2.1.0.1 Événement L'expression la plus élémentaire s'appelle un événement. Les événements sont paramétrés avec une valeur optionnelle. Par exemple, le fait de taper dans la barre de recherche puis de soumettre la recherche en appuyant sur la touche "Entrée" produit l'événement *SubmitResearch$smartphone*. L'événement *SubmitResearch* sans valeur est validé en effectuant une recherche quelconque.

5.2.1.0.2 Opérateur de disjonction L'opérateur *or* permet de spécifier une disjonction entre deux expressions. La règle $SubmitResearchorClickResearchSuggestion$ correspond à l'action de rechercher un produit *or* pour choisir une suggestion proposée par l'application e-commerce. La règle est validée lorsque au moins une de ces expressions est détectée.

5.2.1.0.3 Opérateurs de séquence L'opérateur de séquence spécifie un ordre d'occurrence pour les actions. Deux opérateurs sont pos-

sibles. Le premier exige que le premier événement soit suivi directement par le second. Par exemple, $SubmitResearchAddFilter$ refuse la séquence $SubmitResearchSortProductsAddFilter$. L'opérateur $\Rightarrow$, indique que les deux événements ne doivent pas nécessairement être consécutifs. Par exemple, $SubmitResearch \Rightarrow AddFilter$, accepte la séquence $SubmitResearchSortProductsAddFilter$.

5.2.1.0.4 Opérateur d'itération L'opérateur + indique qu'une expression doit apparaître au moins une fois. La règle $ClickOnProduct+$ exige que le testeur ajoute au moins un produit à son panier. L'opération *[n]* définit un nombre fixe d'itérations, n étant un nombre entier positif.

```
1     ( SubmitResearch or
2     ClickResearchSuggestion or
3     ClickSearchButton ) =>
4   AddFilter$Used =>
5   ClickOnProduct[3] =>
6   ClickCheckout
```

Listing 5.1 – Séquence d'événements de la commande d'un smartphone

Listing 5.1 est un scénario demandant d'ajouter au moins trois produits reconditionnés dans le panier d'achat avant de valider la commande.

5.2.2 Evaluation de la progression du testeur

Nous compilons le scénario en une machine à états qui identifie les séquences d'événements remplissant le scénario de la tâche. Nous exécutons ensuite la machine à états : chaque fois que le travailleur effectue une interaction, nous déclenchons l'événement correspondant. Nous savons alors si le travailleur remplit le scénario. La figure 5.2 montre la machine à états pour notre exemple de tâche consistant à commander un produit après avoir appliqué un filtre. Notez que l'événement *AddFilter* ne spécifie pas

le filtre à choisir. L'application propose plusieurs filtres de produits, tels que les articles ayant reçu une note de trois étoiles de la part des clients ou les produits respectueux de l'engagement climatique. Le choix du filtre permet au testeur d'introduire une certaine variété dans l'exécution de la tâche.

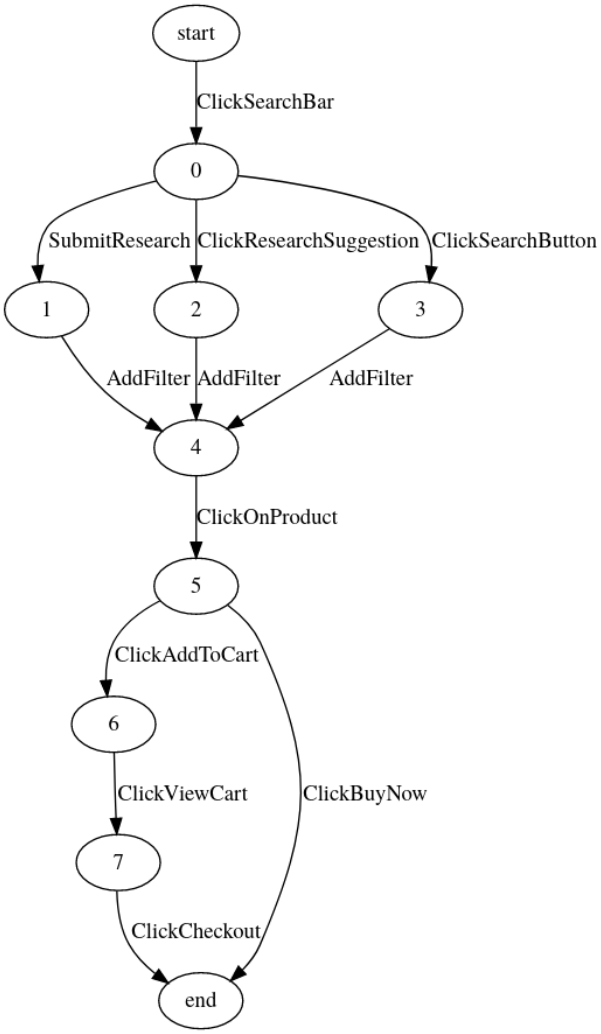


FIGURE 5.2 – Machine à états de l'événement pour la commande d'un produit.

Par exemple, Bob effectue une exploration dans laquelle il recherche un smartphone reconditionné, sélectionne le premier résultat, l'ajoute au panier et appuie sur le bouton de paiement. Il crée la séquence d'événements listée dans le Listing 5.2.

```
1    ClickSearchBar
2    SubmitResearch$smartphone
```

```
3     AddFilter$Used
4     ClickOnProduct$1
5     ClickAddWaranty
6     ClickViewCart
7     ClickCheckout
```

Listing 5.2 – Séquence d'événements de la commande d'un smartphone

La figure 5.3 montre l'évaluation de la séquence d'événements effectuée par Bob. L'événement *ClickAddWaranty* ne fait pas partie du scénario et ne lui permet pas de progresser dans la machine à états. Cependant, l'événement participe à l'exploration du système.

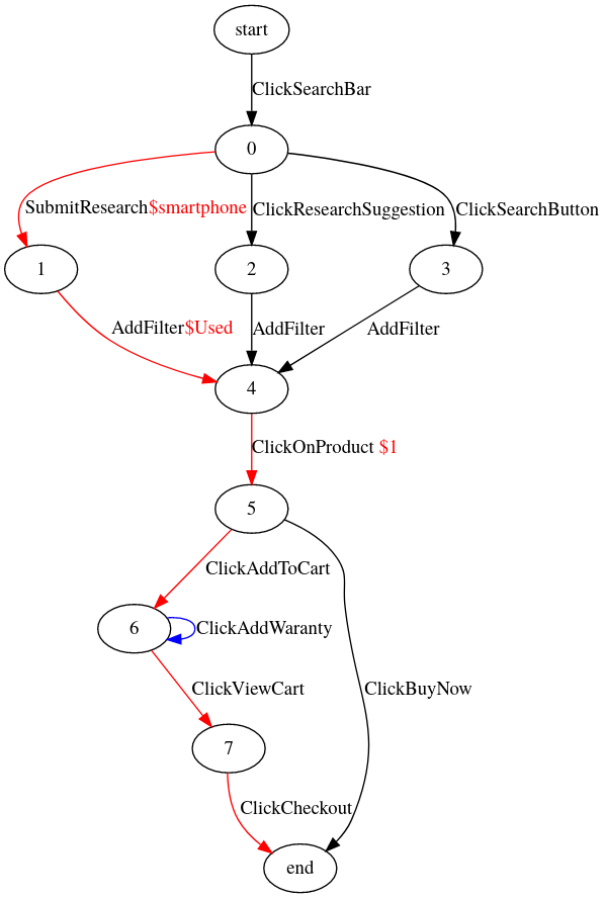


FIGURE 5.3 – Évaluation d'une exploration.

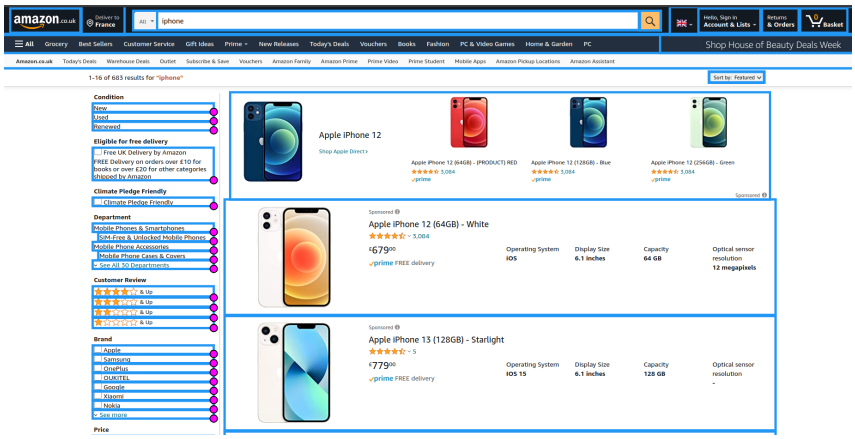
5.2.3 Guider les testeurs dans l'accomplissement de la tâche de test.

Notre technique vise à guider le testeur en lui indiquant quels composants de la page Web lui permettent d'accomplir la tâche de test. Nous utilisons l'état actuel du curseur dans la machine à états pour déterminer la progression du testeur dans le scénario spécifié pour la tâche. Pour l'aider à accomplir la tâche, nous lui montrons comment exécuter les interactions lui permettant de déplacer le curseur dans la machine à états. Pour ce faire, nous récupérons la liste des transitions qui ont pour origine l'état actuel du curseur, puis nous trouvons les éléments de la page web qui permettent à ces événements de se produire.

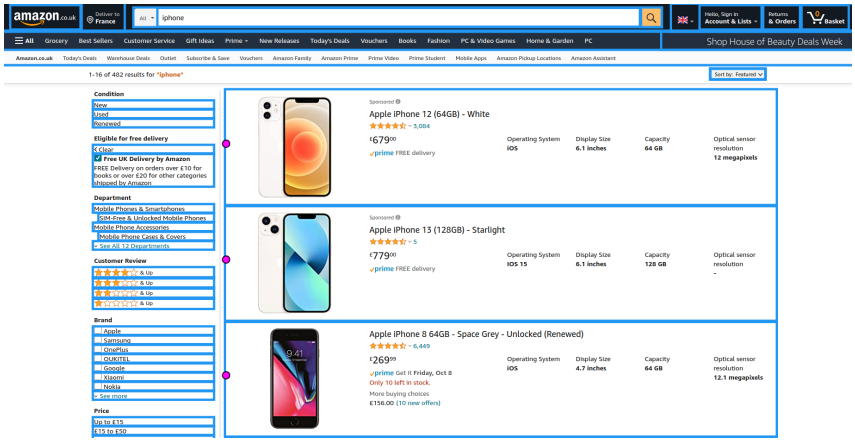
Notez que le but n'est pas d'aller le plus vite possible vers un état final, mais d'aider le testeur à comprendre le périmètre défini pour la tâche. Ainsi, la machine à états peut contenir des boucles, par exemple. Considérons un scénario demandant de mettre au moins un produit dans le panier d'achat. Lorsque le testeur clique sur le premier produit, nous continuons à lui dire qu'il peut effectuer les actions *ClickOnProduct* puis *ClickAddToCart*. Bien que ce cheminement ne fasse pas avancer la tâche de test, il encourage une exploration plus poussée.

Pour indiquer au testeur comment se déplacer dans la machine à états, nous identifions les composants de la page Web qui permettent de déclencher les événements requis directement dans l'interface. Comme les crowdtesters peuvent parfois être très pressés de terminer une tâche, nous avons choisi d'utiliser une animation d'une balle rose cerclée de noir se déplaçant autour des éléments d'intérêt. Le mouvement de la balle attire l'attention du testeur sur l'élément de la page sans ambiguïté.

La Figure 5.4 affiche la page Web que le testeur voit avant (cf : Figure 5.4a) et après (cf : Figure 5.4b) la sélection d'un filtre. La boule est animée pour se déplacer autour de l'élément graphique ; dans la Figure 5.4b, on peut voir qu'il y a une boule pour chaque filtre sélectionné. De cette manière, le testeur comprend la tâche à accomplir et détermine les interactions nécessaires pour la réaliser. Il n'y a aucune confusion entre



a La superposition indique qu'il faut choisir un filtre.



b La superposition indique qu'il faut cliquer sur un produit.

FIGURE 5.4 – Guider le testeur avec la superposition de l'interface graphique

les filtres et les catégories (Épicerie, Meilleures ventes, etc.) sous le bandeau supérieur. Lorsque le testeur sélectionne un filtre, l'événement *AddFilter* est déclenché, le jeton dans la machine à états avance et la superposition est mise à jour pour montrer les prochaines interactions attendues. Les bordures bleues entourant les éléments de la page web montrent ceux qui provoquent un événement enregistré par notre assistant. Nous montrons ces bordures pour informer le testeur de l'interaction enregistrée et éviter tout malentendu.

5.3 Évaluation

Nous décrivons dans cette section l'étude contrôlée que nous avons réalisée pour étudier l'hypothèse selon laquelle notre approche peut aider les crowdtesters à exécuter un scénario sans nuire à la diversité des explorations qu'ils effectuent.

Pour mener notre étude, nous avons choisi d'utiliser deux applications web de production, Amazon et Reddit. Il s'agit de deux applications grand public fournissant des services de nature différente. Amazon est une application de commerce électronique, tandis que Reddit est un réseau social. Les testeurs sont susceptibles d'être déjà familiarisés avec ces applications. Nous avons conçu un scénario simple proche de l'utilisation standard de chacune de ces applications. Ils sont conçus pour être facilement compréhensibles mais nécessitent tout de même de suivre plusieurs étapes. Nous voulons mesurer l'impact de notre approche pour faire suivre aux crowdtesters un scénario légèrement complexe.

5.3.1 Mise en place expérimentale

Nous avons utilisé Amazon Mechanical Turk pour réaliser les sessions de crowdtesting. Nous demandons aux testeurs d'installer notre extension de navigateur et d'entrer l'URL de connexion pour rejoindre la session de test. La participation à la session ouvre une nouvelle page du site Web à explorer. Pour encourager le testeur à suivre correctement le scénario de test, nous avons conditionné la validation des tâches à un mot de passe. Nous avons défini deux mots de passe, un code de participation donné au testeur lors de la saisie de l'URL de connexion dans le plugin. Le code de participation permet de valider la tâche et de recevoir 0.10$. Le second code, appelé code d'achèvement, permet au testeur d'obtenir un bonus de 1 $, soit un total de 1,10 $. Lorsque le serveur renvoie que le testeur a terminé une séquence d'événements validant le scénario, le testeur peut appuyer sur le bouton STOP de notre extension pour obtenir le code d'achèvement. Si le testeur appuie sur le bouton STOP avant de terminer, une popup

l'informe que le scénario est incomplet. Les crowdtesters devaient avoir un taux d'acceptation minimum de 90% pour participer et avoir accompli au moins 200 tâches sur la plateforme.

5.3.1.0.1 Conception des sessions de test Pour mettre en place des tâches de test réalistes, nous avons utilisé des applications web de production : Amazon et Reddit. Nous voulions tester des sites Web couramment utilisés que les testeurs peuvent déjà connaître. Notre approche utilise une correspondance entre les éléments DOM identifiés par un sélecteur CSS et les événements abstraits utilisés pour évaluer l'exploration. Nous avons assez facilement identifié les composants de page d'Amazon et de Reddit car les éléments DOM principaux ont un identifiant unique. De plus, la plupart des principaux composants de page avaient des attributs HTML dédiés aux tests automatisés. Nous avons configuré la plupart des mappages automatiquement en faisant correspondre l'id unique ou l'attribut de test avec les événements. Les configurations utilisées sont disponible en Annexe A . II . et Annexe A . III .. Pour faciliter la compréhension, nous avons renommé les événements correspondant aux principaux composants, tels que les cases à cocher Filtrer sur Amazon ou les boutons Enregistrer sur Reddit. Nous avons ensuite conçu deux scénarios. La figure 5.1 présente les éléments du scénario Amazon, et la figure 5.5 montre les pages Web du scénario Reddit.

- Scénario Amazon
  - Recherche d'un smartphone.
  - Appliquez un filtre en utilisant les cases à cocher ou les étoiles dans le menu de gauche.
  - Ajoutez un ou plusieurs produits au panier.
  - Cliquez sur le bouton "Passer à la caisse".
- Reddit Scenario
  - Sélectionnez l'une des communautés.

— Utilisez le filtre Top pour récupérer les messages les plus votés.

— Enregistrez trois messages à l'aide du bouton save.

— Ouvrez votre page de profil.

— Cliquez sur la section SAVED.

— Annulez l'enregistrement d'un de vos messages.

— Cliquez sur le bouton hide de l'un de vos messages.

— Ouvrez la page HIDDEN.

— Cliquez sur Unhide sous l'un de vos messages.

Nous avons défini quatre lots de 30 tâches, deux sur le site Web d'Amazon et deux sur Reddit.

— A_a et A_{na} pour l'exécution assistée et non assistée du scénario Amazon.

— R_a et R_{na} pour l'exploration assistée et non assistée du scénario Reddit.

Au total, l'étude donne 120 (30 tâches X 4 lots) points de données.

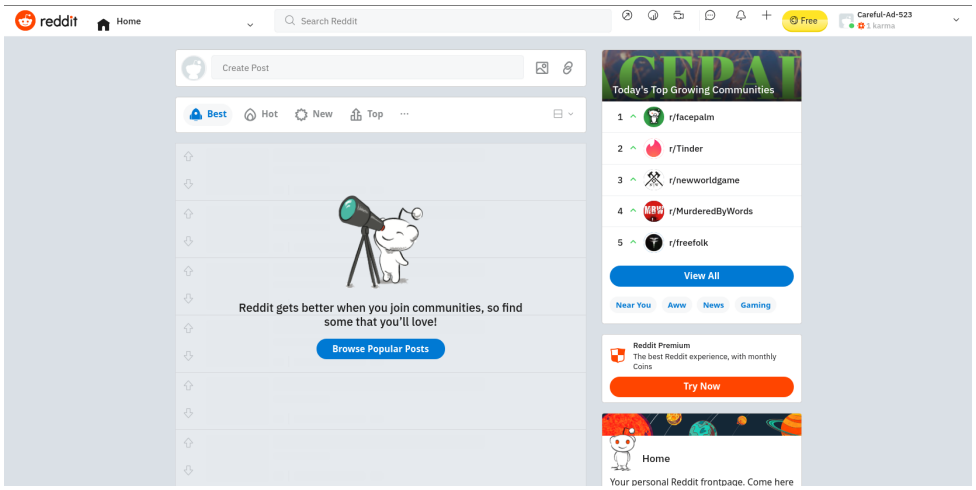
5.3.1.0.2 Métriques mesurées Pour chaque lot, nous avons mesuré le nombre d'exécutions correctes du scénario et la couverture des combinaisons d'événements des explorations valides.

— Explorations valides : nombre d'exécutions correctes du scénario.

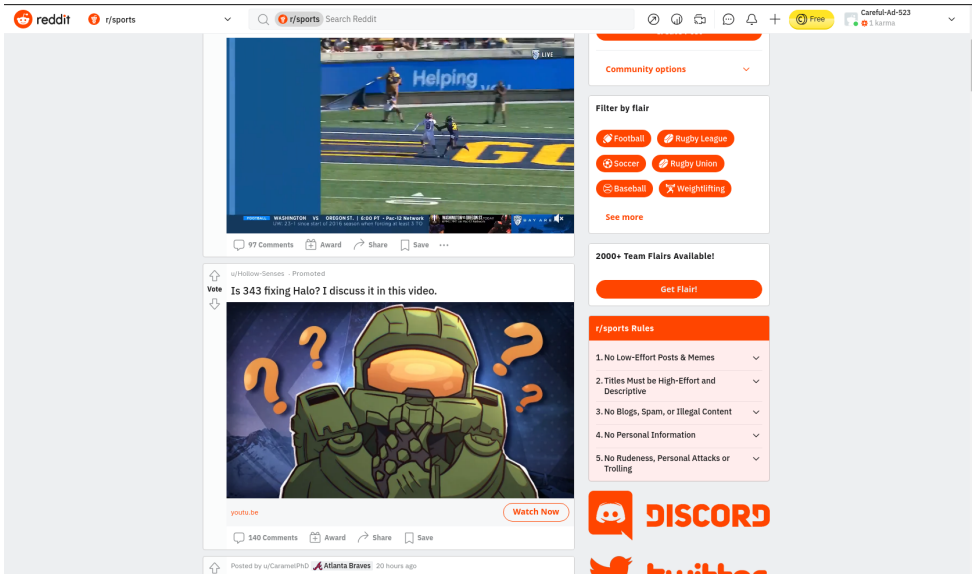
— Couverture de n-gram : nombre de sous-séquences observées de taille 1 à 9. Nous exécutons neuf fenêtres glissantes de taille 1 à 9 sur les séquences d'événements et comptons le nombre d'occurrences de chaque sous-séquence. Cette métrique nous donne le nombre de contextes différents dans lesquels les testeurs ont réalisé les événements.

— Tentatives d'envoi : nombre de fois où le testeur a appuyé sur le bouton STOP.

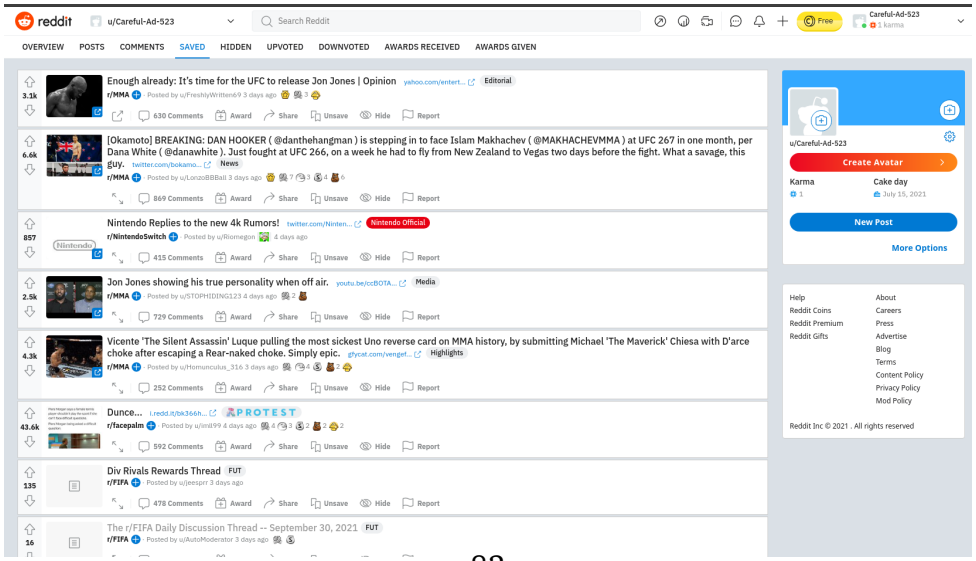
Ces mesures indiquent l'efficacité globale du guidage sur les lots de tâches de test.



a Reddit Home Page.



b posts Reddit.



c Posts enregistrés sur Reddit.

FIGURE 5.5 – Pages explorées sur Reddit

TABLE 5.1 – Résultats de la session sur les 4 lots de 30 tâches.

Session	explorations valides	n-grammes couverts
R_a	15	798
R_{na}	5	273
A_a	14	1375
A_{na}	5	468

5.3.2 Résultats

Nous examinons dans cette section l'effet de notre approche sur le nombre d'exécutions de scénarios valides et le nombre de combinaisons d'interactions couvertes. Pour mesurer la signification statistique, nous avons utilisé un test de Fisher. Les résultats sont présentés dans 5.1.

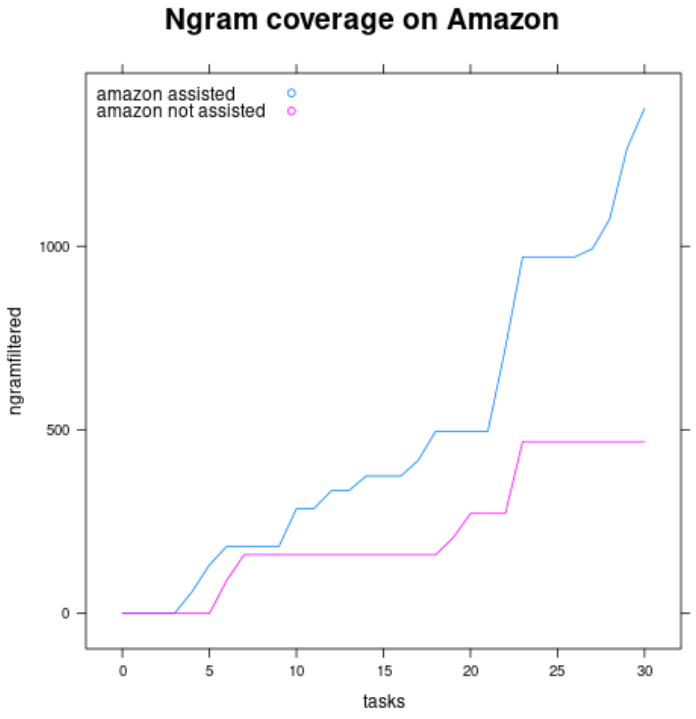


FIGURE 5.6 – n-grams couverts lors de la session de crowd-testing sur Amazon.

RQ1 : Est-ce que les crowdtesters non guidés effectuent des explorations correctes ? *Les testeurs non guidés produisent une grande majorité d'exécutions incorrectes*

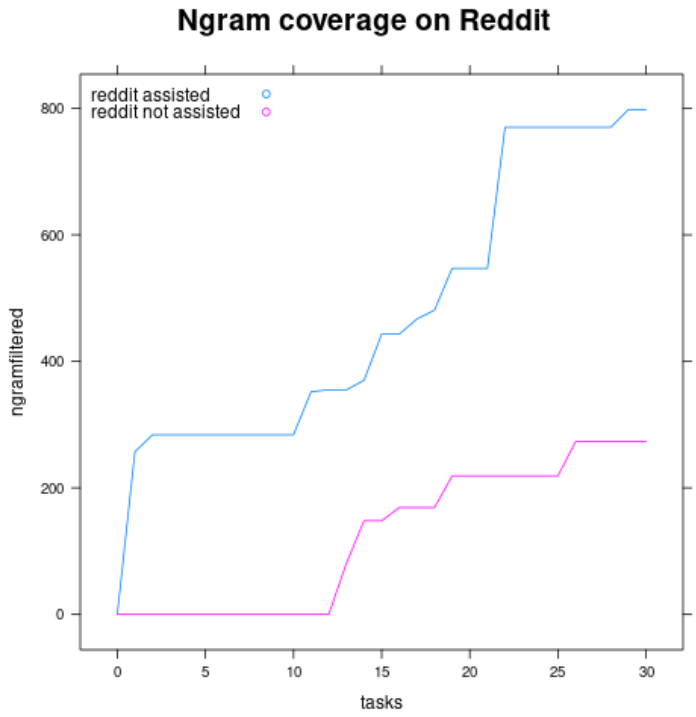


FIGURE 5.7 – n-grams couverts lors de la session de crowd-testing sur Reddit.

du scénario. Les résultats des sessions R_{na} et A_{na} montrent dans les deux cas que les participants ne suivent pas le scénario correctement (83,4%). Dans le cas d'Amazon, nous avons remarqué que les explorations invalides ont tendance à suivre l'entonnoir d'achat, mais exécutent toutes les étapes du scénario. L'erreur la plus fréquente est d'oublier l'ajout d'un filtre à la recherche. Pour les lots Reddit, nous attendons des participants qu'ils ouvrent leur page de profil pour désenregistrer les messages qu'ils ont précédemment sauvegardés. La plupart d'entre eux cliquent correctement sur le bouton de sauvegarde des posts mais n'ouvrent pas la page de profil pour les désenregistrer, sautant cette page, car Reddit leur permet de désenregistrer sans utiliser ce menu. Dans les deux cas, les testeurs sautent une étape qui n'est pas représentative de l'utilisation directe de l'application, ce qui indique qu'ils ne lisent pas et n'exécutent pas le scénario ligne par ligne. Cela renforce l'idée qu'un guidage visuel doit apparaître directement sur la page web.

RQ2 : Les crowdtesters assistés effectuent-ils plus d'explorations correctes que

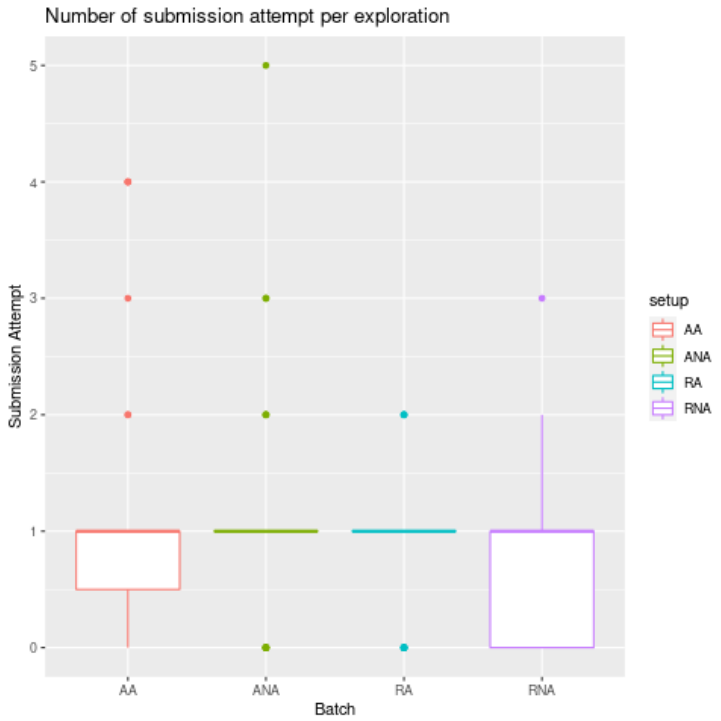


FIGURE 5.8 – Nombre de tentatives d'envoi par tâche.

les non guidés ? *Les testeurs assistés produisent plus d'explorations valides par rapport au scénario.* En comparant les sessions assistées et non assistées, nous avons constaté que les assistés réalisent beaucoup plus d'explorations valides par rapport au scénario. On observe une augmentation de 200La figure 5.8 montre le nombre de tentatives de soumission par tâche. Il n'y a pas de différence significative dans le nombre de tentatives d'envoi. Nous interprétons ce résultat comme une indication que les testeurs abandonnent rapidement s'ils ne font pas une exécution correcte du premier coup.

RQ3 : Les crowdtesters assistés atteignent-ils une couverture plus élevée que les non guidés ? *Les testeurs assistés explorent plus de combinaisons d'interactions.* Nous avons mesuré une augmentation de 192% entre R_a et R_{na}, et une augmentation de 193% entre A_a et A_{na}. Nous nous intéressons à la croissance de la couverture et traçons les Figures 5.6 et Figure 5.7, qui montrent l'évolution de la couverture tâche par tâche. La figure 5.9 montre le nombre de nouveaux n-grams couverts par chaque exploration, nous avons appliqué un test de Mann Whitney pour évaluer l'effet de notre approche

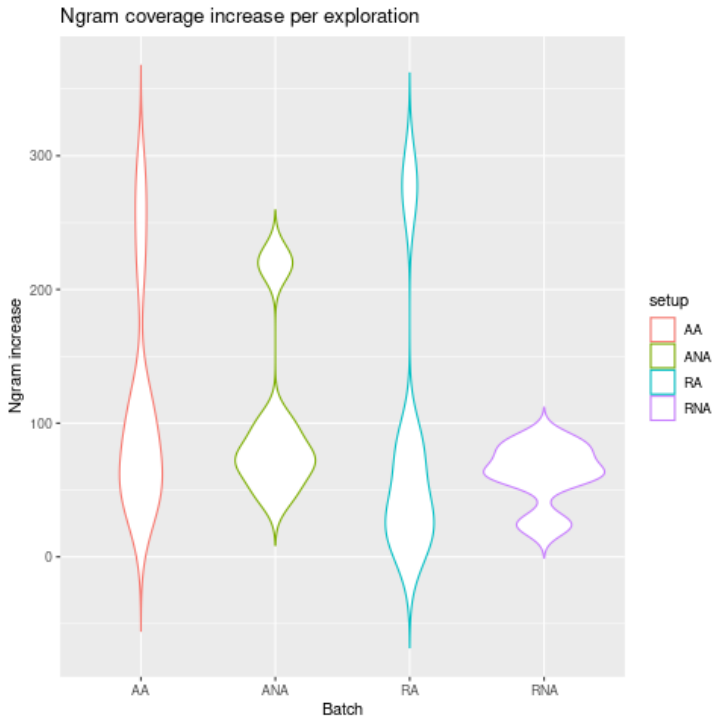


FIGURE 5.9 – Nombre de nouveaux ngrams couverts par exploration.

sur l'augmentation de la couverture par exploration et avons obtenu $\sigma = 0,017320$ sur les lots Amazon, indiquant un effet significatif, alors que l'effet sur les lots Reddit n'était pas significatif $\sigma = 0,084572$.

En résumé, nous avons constaté que nous pouvons guider les testeurs dans l'exécution d'un scénario pour un nombre accru d'exécution valide. Nous avons également constaté que la couverture d'interaction n'est pas lésée par le guidage pas à pas, ce qui se traduit par une meilleure couverture d'interaction globale, puisque davantage d'explorations sont utilisables.

5.3.3 Menaces sur la validité

Nous recrutons des testeurs à l'aide d'Amazon Mechanical Turk et empêchons les travailleurs ayant le même identifiant de participer à la même tâche. Mais nous n'avons pas été en mesure d'empêcher un travailleur de participer à plusieurs tâches. Cela peut

induire un effet d'apprentissage où les testeurs sont déjà familiarisés avec notre outil de guidage tandis que d'autres le découvrent. Comme les résultats d'Amazon et de Reddit sont très similaires, nous pensons que cet effet est limité.

Il y a un code de participation et un code d'achèvement pour chaque lot de tâches. Si les testeurs peuvent communiquer le code, certains d'entre eux peuvent avoir validé la tâche sans vraiment participer. Puisqu'il n'y a aucune preuve que des tâches ont été soumises sans être réalisées (la longueur de l'exploration est cohérente), nous ne pensons pas que le système de code soit abusé.

5.4 Synthèse

Dans ce Chapitre, nous abordons la question de l'intégration du crowdsourcing aux tests exploratoires. Nous proposons un DSL pour définir un scénario de test et une assistance au niveau de l'interface graphique pour guider les crowdtesters. Nous guidons le testeur en utilisant une surcouche graphique superposée à la page Web utilisée par le testeur pendant la session de test. Elle aide les testeurs à identifier les interactions qu'ils peuvent utiliser pour progresser dans l'exécution du scénario. En outre, le scénario autorise un certain degré d'exploration, permettant aux testeurs de personnaliser leur exécution et d'explorer ainsi diverses combinaisons d'interactions. Nos résultats montrent que guider les testeurs pendant l'exécution de la tâche augmente significativement le nombre d'explorations qui suivent les directives. Nous avons également découvert que le guidage étape par étape n'a pas d'effet négatif sur la couverture des interactions, ce qui entraîne une meilleure couverture des interactions en raison de la présence d'explorations supplémentaires.

Chapitre 6

Conclusion et perspectives

Sommaire

I . Résumé

L'objectif général est d'outiller les pratiques de test exploratoire en guidant les testeurs à partir de modélisations, soit construites en amont des sessions de test pour définir un scénario, soit pendant la session en assimilant les interactions des testeurs avec l'application. Ces approches permettent d'assister les testeurs, tout en laissant suffisamment de liberté pour continuer de faire levier sur leurs connaissances.

Notre première contribution est une approche utilisant un Modèle de Markov pour recommander les actions ayant été peu ou pas réalisées par les testeurs, et ainsi leur permettre d'améliorer la diversité des sessions de test exploratoire. Nous construisons un modèle en assimilant les explorations réalisées au cours d'une session de test pour afficher un retour en temps réel. Ainsi les testeurs reçoivent des recommandations basées à la fois sur leurs propres explorations, mais aussi sur celles des autres participants.

Cette approche a été validée dans deux études contrôlées et une étude de cas réalisée avec notre partenaire industriel. Nous avons montré que l'assistance permet d'améliorer la diversité des sessions de test exploratoire.

Notre seconde contribution est une approche pour conduire des sessions de test exploratoire avec des testeurs recrutés via des plateformes de crowdsourcing. Exécuter correctement une tâche de test exploratoire sans avoir de connaissance préalable sur l'application testée est difficile. Pour cette raison, on ne peut attendre de testeurs recrutés via une plateforme de crowdsourcing, de réaliser sans assistance, des tâches de test contribuant fortement à la couverture de la session. L'objectif de cette approche est d'améliorer le nombre de tâches de test correctement effectuées par les testeurs, en les orientant étape par étape dans la tâche de test, tout en leur laissant suffisamment de liberté pour qu'ils puissent explorer de nouveaux chemins. Cette approche a été validée par une étude contrôlée réalisée via une plateforme de crowdsourcing.

Notre troisième contribution est AIFEX, un outil pour enregistrer les explorations produites par les testeurs sur une page web lors des sessions de test. Cet outil se présente sous la forme d'un ensemble de service web chargés de stocker et interpréter les explorations afin de produire des recommandations pour les prochaines actions à réaliser dans le but d'améliorer la qualité des sessions de test exploratoire. En complément, une extension web est chargée de faire la traduction entre les événements JavaScript observés sur la page web et les séquences d'actions formant les explorations des testeurs. L'extension de navigateur envoi les explorations aux services Web et reçoit les recommandations, qu'elle affiche directement dans la page web du testeur. AIFEX a pour objectif de service de socle pour la mise en place d'approche d'assistance pour les testeurs.

Nos travaux montrent que l'accompagnement des sessions de test exploratoire peut amener des améliorations significatives sur la couverture de test. Le sujet nous répond à un réel besoin d'assister le test exploratoire.

II . Discussion et Travaux Futurs

Le contrôle de la qualité des systèmes logiciels est un domaine de recherche important. Ceci est amplifié par le fait que la taille et la complexité des systèmes ne cesse d'augmenter. Nous n'en sommes pas encore au stade où l'intelligence artificielle (IA) vérifierait nos systèmes, à ce moment, l'IA pourrait aussi bien produire le logiciel. En attendant, nous pensons que l'intervention humaine, guidée et assistée par des outils est notre meilleur atout. Passé un certain seuil, l'automatisation a tendance à conduire à une augmentation drastique des coûts et une exigence de rigueur élevée à mesure que l'on tend vers une "automatisation complète". Nous préférons miser sur l'accompagnement des testeurs en faisant levier sur leur créativité et leur connaissance pour la réalisation de tâches difficiles. Nous présentons dans cette section les perspectives pour nos futurs travaux.

Une première perspective à court terme est d'exploiter des traces issues de session d'utilisateurs. Dans nos échanges avec l'industrie, il apparait qu'identifier le comportement réel des utilisateurs sur le système est une problématique qui intéresse les entreprises. Cependant, tous les chemins ne sont pas forcément d'égale importance et un testeur pourra décider de ne pas explorer un chemin de navigation, s'il juge que celui-ci n'a que peu de chance de déclencher un bug, ou qu'un bug déclenché de cette façon n'impactera pas la valeur de l'application. Des travaux comme celles proposées par Memon et Brooke [13], Herbold [27], Sprenkle et al. [53] utilisent les séquences d'actions réalisées par les utilisateurs finaux pour identifier les interactions les plus fréquemment réalisées, et donc importante d'un point de vue commerciale. L'approche développée dans le Chapitre 5, permet, pour un scenario donné, d'obtenir des traces réalisées par des crowdtesters. De cette façon, il est possible d'obtenir un profil d'utilisation. Notre objectif est d'assister la rédaction de suite de test, en comparant les séquences d'actions d'un scenario de test, avec l'utilisation réelle du système décrite dans le profile d'utilisation.

Deuxièmement, le développement des approches présentées dans cette thèse ont

menées à la construction d'AIFEX, un socle logiciel open source pour accompagner les pratiques de test exploratoire. AIFEX a été développé avec l'objectif de permettre l'ajout de nouvelles façons de guider les testeurs, et ainsi servir de socle expérimental. Notre objectif est d'adapter AIFEX pour le rendre utilisable dans un contexte industriel, en ainsi servir de passerelle d'échange entre le monde de la recherche, et celui de l'industrie sur le sujet du test des applications web.

Troisièmement, reproduire un bug est un problème complexe, il faut définir dans un système complexe, le contexte qui a permis de déclencher le bug. Le système de recommandation que nous proposons utilise les $n-1$ actions précédentes pour décrire ce contexte et recommander la prochaine action à réaliser. Au cours de cette thèse, nous avons implémenté des systèmes de recommandations basés sur les *Item Set* et les *Sequence Pattern*, cependant Quadrana et al. [49] pointent d'autres approches que nous n'avons pas évaluées. De plus la modélisation que nous avons choisie pour les actions n'a pas été comparée à d'autres approches. Notre modélisation n'utilise qu'une sous partie des données disponibles, et nous pensons qu'inclure plus d'informations dans les modélisations, comme la localisation, la langue, la résolution de l'écran, peut enrichir la qualité des recommandations. Nous pensons qu'un important travail d'évaluation des modélisations est nécessaire. Ainsi l'élaboration d'un système de recommandation pour les tests système est un sujet a plus long terme.

Bibliographie

[1] internetlivestats.

[2] Selenium.

[3] State of Agile survey.

[4] W. Afzal, A. N. Ghazi, J. Itkonen, R. Torkar, A. Andrews, and K. Bhatti. An experiment on the effectiveness and efficiency of exploratory testing. *Empirical Software Engineering*, 20(3) :844–878, June 2015.

[5] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar. Quality Control in Crowdsourcing Systems : Issues and Directions. *IEEE Internet Computing*, 17(2) :76–81, 2013.

[6] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move : user activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the 15th international conference on World Wide Web*, pages 203–212, 2006.

[7] J. Bach. Session-based test management. *Software Testing and Quality Engineering Magazine*, 2(6) :32–37, 2000.

[8] A. Beer and R. Ramler. The role of experience in software testing practice. In *2008 34th Euromicro Conference Software Engineering and Advanced Applications*, pages 258–265. IEEE, 2008.

[9] F. Belli, C. J. Budnik, and L. White. Event-based modelling, analysis and testing of user interactions : approach and case study. *Software Testing,*

Verification and Reliability, 16(1) :3–32, 2006. _eprint : https ://onlinelibrary.wiley.com/doi/pdf/10.1002/stvr.335.

[10] K. Bhatti and A. N. Ghazi. *Effectiveness of Exploratory Testing, An empirical scrutiny of the challenges and factors affecting the defect detection efficiency*. 2010.

[11] J. M. Bland and D. G. Altman. The logrank test. *Bmj*, 328(7447) :1073, 2004. Publisher : British Medical Journal Publishing Group.

[12] W. Braik. *Détection d'évènements complexes dans les flux d'évènements massifs*. PhD Thesis, Bordeaux, 2017.

[13] P. A. Brooks and A. M. Memon. Automated GUI testing guided by usage profiles. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*, pages 333–342, 2007.

[14] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based N-gram Models of Natural Language. *Comput. Linguist.*, 18(4) :467–479, Dec. 1992.

[15] B. Burg, R. Bailey, A. J. Ko, and M. D. Ernst. Interactive Record/Replay for Web Application Debugging. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 473–484, New York, NY, USA, 2013. ACM.

[16] Y. Chen, M. Pandey, J. Song, W. Lasecki, and S. Oney. Improving Crowd-Supported GUI Testing with Structural Guidance. pages 1–13, 2020.

[17] T. S. Chow. Testing software design modeled by finite-state machines. *IEEE transactions on software engineering*, (3) :178–187, 1978. Publisher : IEEE.

[18] J. M. Clarke. Automated test generation from a behavioral model. In *Proceedings of Pacific Northwest Software Quality Conference. IEEE Press*. Citeseer, 1998.

[19] F. Cornelis, A. Georges, M. Christiaens, M. Ronsse, T. Ghesquiere, and K. Bosschere. A taxonomy of execution replay systems. In *Proceedings of International*

Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet. Citeseer, 2003.

[20] E. Dolstra, R. Vliegendhart, and J. Pouwelse. Crowdsourcing GUI Tests. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, pages 332–341, 2013.

[21] P. Ellis. Thresholds for interpreting effect sizes. *Retrieved January*, 13 :2014, 2009.

[22] R. Gao, Y. Wang, Y. Feng, Z. Chen, and W. Eric Wong. Successes, challenges, and rethinking – an industrial investigation on crowdsourced mobile application testing. *Empirical Software Engineering*, 24(2) :537–561, Apr. 2019.

[23] A. N. Ghazi, R. Garigapati, and K. Petersen. Checklists to Support Test Charter Design in Exploratory Testing. Apr. 2017.

[24] A. N. Ghazi, K. Petersen, E. Bjarnason, and P. Runeson. Levels of Exploration in Exploratory Testing : From Freestyle to Fully Scripted. *IEEE Access*, 6 :26416–26423, 2018.

[25] V. H. M. Gomide, P. A. Valle, J. O. Ferreira, and J. R. G. Barbosa. Affective Crowdsourcing Applied to Usability Testing. 5 :5, 2014.

[26] D. R. Hackner and A. M. Memon. Test case generator for GUITAR. In *Companion of the 30th international conference on Software engineering*, pages 959–960, 2008.

[27] S. Herbold. *Usage-based Testing of Event-driven Software*. PhD Thesis, Niedersächsische Staats-und Universitätsbibliothek Göttingen, 2012.

[28] T. Hoßfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia. Best Practices for QoE Crowdtesting : QoE Assessment With Crowdsourcing. *IEEE Transactions on Multimedia*, 16(2) :541–558, 2014.

[29] J. Itkonen. Do test cases really matter ? An experiment comparing test case based and exploratory testing. 2011.

[30] J. Itkonen, M. V. Mantyla, and C. Lassenius. Defect Detection Efficiency : Test Case Based vs. Exploratory Testing. In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, pages 61–70, Sept. 2007.

[31] J. Itkonen and K. Rautiainen. Exploratory testing : a multiple case study. In *2005 International Symposium on Empirical Software Engineering, 2005.*, pages 82–91, Queensland, Australia, 2005. IEEE.

[32] C. Kaner, J. Falk, and H. Q. Nguyen. *Testing computer software*. John Wiley & Sons, 1999.

[33] E. L. Kaplan and P. Meier. Nonparametric Estimation from Incomplete Observations. *Journal of the American Statistical Association*, 53(282) :457–481, 1958. Publisher : [American Statistical Association, Taylor & Francis, Ltd.].

[34] D. J. Kasik and H. G. George. Toward automatic generation of novice user test scripts. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 244–251, 1996.

[35] D. G. Kleinbaum and M. Klein. *Survival analysis : a self-learning text*. Statistics for biology and health. Springer, New York, NY, 3rd ed. edition, 2012. Publication Title : Survival analysis : a self-learning text.

[36] N. Leicht, I. Blohm, and J. M. Leimeister. Leveraging the power of the crowd for software testing. *IEEE Software*, 34(2) :62–69, 2017. Publisher : IEEE.

[37] G. Leshed, E. Haber, and T. Matthews. CoScripter : Automating & Sharing How-To Knowledge in the Enterprise. pages 1719–1728, 2008.

[38] J. Leveau, X. Blanc, J.-R. Falleri, L. Réveillère, and R. Rouvoy. Fostering the Diversity of Exploratory Testing in Web Applications. In *13th IEEE Conference on Software Testing, Validation and Verification,ICST 2020, Porto, Portugal, October 24-28, 2020*. IEEE, 2020.

[39] L. Mariani, M. Pezze, O. Riganelli, and M. Santoro. Autoblacktest : Automatic black-box testing of interactive applications. In *2012 IEEE Fifth Internatio-*

nal Conference on Software Testing, Verification and Validation, pages 81–90. IEEE, 2012.

[40] P. M. Maurer. Generating test data with enhanced context-free grammars. *Ieee Software*, 7(4) :50–55, 1990. Publisher : IEEE.

[41] A. M. Memon. An event-flow model of GUI-based applications for testing. *Softw. Test., Verif. Reliab.*, 17(3) :137–157, 2007.

[42] A. M. Memon, M. L. Soffa, and M. E. Pollack. Coverage criteria for GUI testing. In *Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 256–267, 2001.

[43] A. Mesbah and A. v. Deursen. Migrating Multi-page Web Applications to Single-page AJAX Interfaces. In *11th European Conference on Software Maintenance and Reengineering (CSMR'07)*, pages 181–190, Mar. 2007.

[44] A. Mesbah, A. v. Deursen, and S. Lenselink. Crawling Ajax-Based Web Applications through Dynamic Analysis of User Interface State Changes. *TWEB*, 6(1) :3, 2012.

[45] M. Micallef, C. Porter, and A. Borg. Do Exploratory Testers Need Formal Training ? An Investigation Using HCI Techniques. In *2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 305–314, 2016.

[46] J. Mickens, J. Elson, and J. Howell. Mugshot : Deterministic Capture and Replay for Javascript Applications. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 11–11, Berkeley, CA, USA, 2010. USENIX Association. event-place : San Jose, California.

[47] F. F.-H. Nah. A study on tolerable waiting time : how long are Web users willing to wait ? *Behav. Inf. Technol.*, 23(3) :153–163, 2004.

[48] D. Pfahl, H. Yin, M. V. Mäntylä, and J. Münch. How is Exploratory Testing Used ? A State-of-the-practice Survey. In *Proceedings of the 8th ACM/IEEE International*

Symposium on Empirical Software Engineering and Measurement, ESEM '14, pages 5 :1–5 :10, New York, NY, USA, 2014. ACM. event-place : Torino, Italy.

[49] M. Quadrana, P. Cremonesi, and D. Jannach. Sequence-Aware Recommender Systems. *ACM Comput. Surv.*, 51(4), July 2018. Place : New York, NY, USA Publisher : Association for Computing Machinery.

[50] R. Rosenthal. Parametric measures of effect size. In *The handbook of research synthesis.*, volume 621, pages 231–244. Russell Sage Foundation, New York, NY, US, 1994.

[51] N. Salkind. Encyclopedia of Research Design, 2020.

[52] K. Sen, S. Kalasapur, T. Brutch, and S. Gibbs. Jalangi : A Selective Record-Replay and Dynamic Analysis Framework for JavaScript. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 488–498, New York, NY, USA, 2013. Association for Computing Machinery. event-place : Saint Petersburg, Russia.

[53] S. Sprenkle, L. Pollock, and L. Simko. A Study of Usage-Based Navigation Models and Generated Abstract Test Cases for Web Applications. In *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, pages 230–239, Mar. 2011.

[54] P. Tonella, R. Tiella, F. B. Kessler, and C. D. Nguyen. Interpolated N-Grams for Model Based Testing.

[55] A. Von Mayrhauser and S. Crawford-Hines. Automated testing support for a robot tape library. In *Proceedings of 1993 IEEE International Symposium on Software Reliability Engineering*, pages 6–14. IEEE, 1993.

[56] J. A. Whittaker. *Markov chain techniques for software testing and reliability analysis*. PhD Thesis, The University of Tennessee, 1992.

[57] J. A. Whittaker and M. G. Thomason. A Markov chain model for statistical software testing. *IEEE Transactions on Software engineering*, 20(10) :812–824, 1994. Publisher : IEEE.

[58] Q. Xie and A. M. Memon. Using a pilot study to derive a GUI model for automated testing. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 18(2) :1–35, 2008. Publisher : ACM New York, NY, USA.

[59] X. Yuan, M. B. Cohen, and A. M. Memon. GUI Interaction Testing : Incorporating Event Context. *IEEE Transactions on Software Engineering*, 37(4) :559–574, July 2011.

Annexes

Sommaire

A . Configurations AIFEX des sites web

Nous donnons ici les Règles de mappages utilisées pour nos études controllées sur les sites de CDiscount, Amazon et Reddit. Les règles présentées ici sont un sous ensemble du des règles utilisées, contenant les règles les plus pertinentes.

A . I . CDiscount

```
[
    {
        "match": {
            "event": "click",
            "css": "input.btGreen.btS.jsValidForm",
        },
```

```
7           "output": {
8               "prefix": "AddToBasket",
9               "suffix": "index",
10          }
11      },
12      {
13          "match": {
14              "event": "click",
15              "css": "#hBskt",
16          },
17          "output": {
18              "prefix": "BasketButtonClick",
19          }
20      },
21      {
22          "match": {
23              "event": "change",
24              "css": "#facetsList label",
25          },
26          "output": {
27              "prefix": "FilterClick",
28              "suffix": "index",
29          }
30      },
31      {
32          "match": {
33              "event": "click",
34              "css": "#lpBloc img.prdtBImg",
35          },
36          "output": {
```

```
37              "prefix": "ItemPictureClick",
38              "suffix": "index",
39          }
40      },
41      {
42          "match": {
43              "event": "click",
44              "css": "div.prdtBILDetails > a",
45          },
46          "output": {
47              "prefix": "ItemTitleClick",
48              "suffix": "index",
49          }
50      },
51      {
52          "match": {
53              "event": "click",
54              "css": ".bSummaryOrderLink",
55          },
56          "output": {
57              "prefix": "OrderClick",
58          }
59      },
60      {
61          "match": {
62              "event": "change",
63              "css": "div.bProductLineDescBottomQuantity.
                    jsBProductLineDescBottomQuantity > select",
64          },
65          "output": {
```

```
66              "prefix": "QuantitySelect",
67              "suffix": "value",
68          }
69      },
70      {
71          "match": {
72              "event": "click",
73              "css": "div.bProductLineDescBloc > div.
                    bProductLineDescBottom > form > span:nth-child(7)",
74          },
75          "output": {
76              "prefix": "RemoveItemClick",
77              "suffix": "index",
78          }
79      },
80      {
81          "match": {
82              "event": "click",
83              "css": ".hSrcInput > input",
84          },
85          "output": {
86              "prefix": "SearchClick",
87          }
88      },
89      {
90          "match": {
91              "key": "Enter",
92              "event": "keyup",
93              "css": ".hSrcInput > input",
94          },
```

```
95          "output": {
96              "prefix": "Search",
97              "suffix": "value",
98          }
99      },
100     {
101         "match": {
102             "event": "click",
103             "css": ".hSrcComp > ul > li",
104         },
105         "output": {
106             "prefix": "SearchSelect",
107             "suffix": "index",
108         }
109     },
110     {
111         "match": {
112             "event": "click",
113             "css": "#hFull > div.hSearch > div.hSrcInput > button",
114         },
115         "output": {
116             "prefix": "SearchButtonClick",
117         }
118     },
119     {
120         "match": {
121             "event": "click",
122             "css": ".btGreen.btF",
123         },
124         "output": {
```

```
125             "prefix": "ShowBasket",
126         }
127     },
128     {
129         "match": {
130             "event": "click",
131             "css": "#lpBloc div.prdtBILDetails > div.prdtBILDesc.
                    jsPrdtBILLink",
132         },
133         "output": {
134             "prefix": "ShowDescription",
135             "suffix": "index",
136         }
137     },
138     {
139         "match": {
140             "event": "change",
141             "css": "#lpSort > form > select",
142         },
143         "output": {
144             "prefix": "ChangeSortMode",
145             "suffix": "value",
146         }
147     },
148     {
149         "match": {
150             "event": "change",
151             "css": "#lpBloc div.prdtBILDetails div.prdtBILTwoSel
                    select",
152         },
```

```
153         "output": {
154             "prefix": "ChangeSize",
155             "suffix": "value"
156         }
157     }
158 ]
```

A . II . Reddit

```
1   [
2     {
3         "match": {
4             "css": "[name='createPost']",
5             "event": "click"
6         },
7         "output": {
8             "prefix": "ClickCreatePost"
9         }
10    },
11    {
12        "match": {
13            "css": "._1JNzvBgvzSnX27gUBKqqmJ",
14            "event": "click"
15        },
16        "output": {
17            "prefix": "AccountAction",
18            "suffix": "innerText"
19        }
20    },
21    {
```

```
22          "match": {
23              "css": "._1YWXCINvcuU7nk0ED-bta8",
24              "event": "click"
25          },
26          "output": {
27              "prefix": "AccountMenuOption",
28              "suffix": "innerText"
29          }
30      },
31      {
32          "match": {
33              "css": "[role='search']",
34              "event": "submit"
35          },
36          "output": {
37              "prefix": "Search",
38              "suffix": "value"
39          }
40      },
41      {
42          "match": {
43              "css": "[data-testid]",
44              "event": "click",
45              "attributeName": "data-testid"
46          },
47          "output": {
48              "prefix": "data-testid",
49              "suffix": "attributeValue"
50          }
51      },
```

```
52      {
53          "match": {
54              "css": "[aria-label]",
55              "event": "click",
56              "attributeName": "aria-label"
57          },
58          "output": {
59              "prefix": "aria-label",
60              "suffix": "innerText"
61          }
62      },
63      {
64          "match": {
65              "css": "#ListingSort--Overflow",
66              "event": "click"
67          },
68          "output": {
69              "prefix": "ListingSort--Overflow"
70          }
71      },
72      {
73          "match": {
74              "css": "[role='menuitem']",
75              "event": "click"
76          },
77          "output": {
78              "prefix": "menuitem"
79          }
80      },
81      {
```

```
82          "match": {
83              "css": "#USER_DROPDOWN_ID",
84              "event": "click"
85          },
86          "output": {
87              "prefix": "USER_DROPDOWN_ID"
88          }
89      }
90  ]
```

A . III . Amazon

```
1
2  [
3      {
4        "match": {
5          "css": "\#nav-search-bar-form",
6          "event": "submit"
7        },
8        "output": {
9          "prefix": "searchSubmit"
10       }
11     },
12     {
13       "match": {
14         "css": "\#attachSiAddCoverage",
15         "event": "click"
16       },
17       "output": {
18         "prefix": "addWarrantyToBasketSideMenu"
```

```
19        }
20      },
21      {
22        "match": {
23          "css": "\#attachSiNoCoverage",
24          "event": "click"
25        },
26        "output": {
27          "prefix": "refuseWarranty"
28        }
29      },
30      {
31        "match": {
32          "css": "\#submit.add-to-cart-ubb",
33          "event": "click"
34        },
35        "output": {
36          "prefix": "addToCart"
37        }
38      },
39      {
40        "match": {
41          "css": "\#attach-sidesheet-checkout-button",
42          "event": "click"
43        },
44        "output": {
45          "prefix": "sideButtonCheckout"
46        }
47      },
48      {
```

```
49        "match": {
50          "css": "\#sc-buy-box-ptc-button",
51          "event": "click"
52        },
53        "output": {
54          "prefix": "ProceedToCheckout"
55        }
56      },
57      {
58        "match": {
59          "css": "\#buyNow_feature_div",
60          "event": "click"
61        },
62        "output": {
63          "prefix": "buyNow_feature_div"
64        }
65      },
66      {
67        "match": {
68          "css": "\#addToCart_feature_div",
69          "event": "click"
70        },
71        "output": {
72          "prefix": "addToCart_feature_div"
73        }
74      },
75      {
76        "match": {
77          "css": "\#buyNow_feature_div",
78          "event": "click"
```

```
79        },
80        "output": {
81          "prefix": "buyNow"
82        }
83      },
84      {
85        "match": {
86          "css": "\#add-to-cart-button",
87          "event": "click"
88        },
89        "output": {
90          "prefix": "addToCart"
91        }
92      },
93      {
94        "match": {
95          "css": "\#s-refinements li",
96          "event": "click"
97        },
98        "output": {
99          "prefix": "addFilter",
100         "suffix": "index"
101       }
102     },
103     {
104       "match": {
105         "css": ".s-suggestion",
106         "event": "click"
107       },
108       "output": {
```

```
109         "prefix": "suggestion",
110         "suffix": "index"
111       }
112     },
113     {
114         "match": {
115             "event": "click",
116             "css": "[role='menuitem']",
117             "_id": "6155bf6808b192ffacf92a67"
118         },
119         "output": {
120             "prefix": "menuitem",
121             "_id": "6155bf6808b192ffacf92a68"
122         }
123     },
124     {
125         "match": {
126             "event": "click",
127             "css": "#USER_DROPDOWN_ID",
128             "_id": "6155bf6808b192ffacf92a6a"
129         },
130         "output": {
131             "prefix": "USER_DROPDOWN_ID",
132             "_id": "6155bf6808b192ffacf92a6b"
133         }
134     }
135 ]
```

B . Grammaire du DSL - Scénarios de test

Nous donnons ici la grammaire décrivant notre DSL permettant de construire des scenarios de test exploratoire et utilisée dans le Chapitre 5. La grammaire est écrite pour ANTLR(ANother Tool for Language Recognition).

```
grammar ScenarioGrammar;
OR: 'or';
ARROW: '=>';
IF: 'if';
ELSE: 'else';
NOT: '!';
PLUS: '+';

fragment DIGIT
    :   ('0'..'9');

INTEGER_NUMBER
    :   DIGIT+;

SINGLE_STRING
    : '\'' ~('\'')+ '\''
    ;

DOUBLE_STRING
    : '"' ~('"')+ '"'
    ;

ID : [a-zA-Z0-9_\-]+ ;              // match lower-case identifiers

WS : [ \t\r\n]+ -> skip ; // skip spaces, tabs, newlines
```

```
26
27 action:
28     ID
29     | ID '$' SINGLE_STRING
30     | ID '$' DOUBLE_STRING
31     | ID '$' ID
32     ;
33
34 iteration:
35   '[' INTEGER_NUMBER ']'
36   ;
37
38 string
39     : SINGLE_STRING
40     | DOUBLE_STRING
41     ;
42
43 par_step:
44     '(' step ')'
45     ;
46
47 step:
48     action
49     | step step
50     | step OR step
51     | NOT step
52     | step PLUS
53     | step iteration
54     | step ARROW step
55     | par_step
```

```
56      ;
57
58  main:
59      step EOF;
```