Rapport de stage

Tests de régressions : conception et réalisation

2014 - 2015

Réalisé par Léo Cassiau
Maître de stage : Jean-François Lamballe
Encadrant universitaire : René Thoraval
Licence 3 mention Informatique

UNIVERSITÉ DE NANTES

Remerciements

Mes remerciements s'adressent dans un premier temps à toute l'équipe pédagogique de l'UFR sciences et techniques responsables de la licence Informatique, pour avoir assuré la partie théorique de celle-ci.

Il n'est jamais facile pour un étudiant de trouver un stage, c'est pourquoi je remercie l'entreprise E-Doceo, et plus particulièrement mon maitre de stage, M. Jean-François Lamballe, de m'avoir accueilli au sein d'E-Doceo.

Je tiens à remercier M. Genou pour le temps qu'il m'a consacré tout au long de mon stage, et pour les conseils et avis qu'il m'a prodigués.

Enfin, je remercie toute l'équipe développeur d'E-Doceo pour son accueil et pour leurs nombreuses aides et conseils.

Table des matières

1 Introduction

Dans le cadre de ma 3e année de licence informatique, j'ai effectué un stage de trois mois au sein d'E-Doceo, un éditeur de logiciels. M'intéressant de près à l'éducation, et notamment aux différentes approches pédagogiques existantes, j'ai choisi d'effectuer ce stage au sein d'une entreprise qui a une approche moderne de l'apprentissage. De plus, travailler au sein d'un éditeur de logiciels me permettait de découvrir les réalités des métiers de l'informatique, et d'élargir mes compétences dans le domaine professionnel.

E-Doceo édite des outils permettant à des instructeurs de réaliser des formations afin de les mettre à disposition d'apprenants. Ces outils sont principalement utilisés par des entreprises, celle-ci souhaitant créer ses propres formations destinées à leurs salariés, afin de les former sur des concepts spécifiques à leur secteur d'activité. Ces formations peuvent prendre différentes formes, allant de la simple diapositive commentée au serious game plus ludique. L'un des principaux objectifs d'E-Doceo est donc de répondre aux multiples attentent des entreprises, et notamment d'assurer le bon fonctionnement de leurs logiciels. Dans le cadre de mon stage, j'ai intégré l'équipe de développeurs qui conçoit et perfectionne ces logiciels, afin de repenser et réaliser des tests permettant de prévenir l'apparition de bogues.

Ma mission était donc de proposer de nouvelles approches pour la conception des tests de régression, puis d'en écrire. En effet, élaborer des tests fonctionnels s'avérait laborieux et leur réalisation n'était pas standardisée au sein de l'équipe. L'objectif était donc multiple :

- Créer des outils utilisables dans le futur par l'équipe de développeurs afin d'accélérer la création des tests.
- Répondre à la question suivante : " quand réaliser un test ? ". L'idéal serait de faire un test pour chaque fonctionnalité, mais cela s'avère impossible au vu du nombre de tests à réaliser. Il est donc nécessaire de trouver une alternative réaliste.
- Définir les bonnes pratiques à adopter parmi les nombreuses méthodes facilitant et structurant les tests fonctionnels.
- Standardiser la conception des tests afin de les rendre plus lisibles et faciles à identifier, notamment en définissant la granularité des tests. Faut-il plusieurs tests courts ou un test très long ?

Ce document résume le déroulement des deux premiers mois de mon stage au sein d'E-Doceo, en commençant par présenter la société et les logiciels qu'elle édite. Puis, la mission de mon stage est décrite avec son contexte et ses enjeux. Ensuite, c'est le travail effectué au cours de ces deux mois et les solutions apportées qui sont exposées. Enfin, un bilan professionnel ainsi que personnel clôture ce rapport.

2 E-Doceo, spécialiste du e-learning

2.1 Présentation d'E-Doceo

La société E-Doceo est dirigée par le directeur général Jérôme Bruet. Le siège d'E-Doceo regroupe plus de 80 employés et la compagnie possède maintenant 14 filiales à l'étranger. E-doceo c'est également 5 millions d'utilisateurs, 2300 membres du club utilisateurs, et plus de 1000 clients à travers le monde. Mais E-Doceo est avant tout une société spécialisée dans le digital learning, une nouvelle manière d'apprendre. En effet, E-Doceo édite des logiciels permettant de concevoir du contenu pédagogique, et E-Doceo est également prestataire de services, la compagnie diffusant ces contenus.

FIGURE 1 – Dates clefs de la société E-Doceo
Source : E-Doceo - 03/06/2015

2.2 Le e-learning

L'e-learning désigne l'ensemble des moyens d'apprentissage utilisant les nouvelles technologies, comme l'ordinateur mais également les smartphones ou les tablettes. L'e-learning est principalement destiné aux entreprises qui souhaitent former leurs salariés durablement. En effet, l'e-learning s'avère être particulièrement efficace pour apprendre sur le long terme, tandis que le présentiel est plus efficace au démarrage d'une formation. Associer les deux pratiques afin de profiter de leurs avantages respectifs est une approche appelé le digital learning. Né avec l'apparition d'internet, le digital learning peut prendre plusieurs formes : classe virtuelle, serious game, rapid learning, kit présentiel, etc. L'e-learning n'a donc pas pour vocation de remplacer les méthodes pédagogiques habituelles, mais de les compléter, dans le but de proposer des outils pédagogiques adaptés aux entreprises.

2.3 Les logiciels édités

L'offre d'E-Doceo est aujourd'hui axé sur deux principaux logiciels : le LCMS, E-Doceo Content Manager, qui permet aux instructeurs de créer et de stocker du contenu pédagogique, tandis que le LMS, E-Doceo Learning Manager, permet de diffuser ce contenu aux apprenants. Ces deux logiciels se complètent donc dans le but de faciliter l'accès aux formations, aussi bien pour les formateurs qui souhaitent apporter des modifications, que pour les apprenants à qui elles sont destinées.

2.3.1 E-Doceo Content Manager

Aussi appelé ECMG, E-Doceo Content Manager est composé d'un logiciel et d'une plateforme web, tous les deux destinés aux instructeurs. Le logiciel permet de créer un parcours de formation, dont le contenu est peut-être du rapid learning, un kit présentiel ou une formation scénarisée. La plateforme web quant à elle permet de stocker, modifier et organiser les parcours de formation ainsi que de les diffuser sur ELMG.

2.3.2 E-Doceo Learning Manager

Aussi appelé ELMG, E-Doceo Learning Manager est une plateforme web destiné aux apprenants. ELMG permet de diffuser et d'organiser du contenu pédagogique, de suivre l'évolution des apprenants et de les administrer. ELMG est également complété par un logiciel nommé Catalogue, aussi appelé ETS, qui offre la possibilité aux apprenants d'acheter des formations.

FIGURE 2 – E-Doceo Learning Manager : le LMS d'E-Doceo
Source : E-Doceo - 03/06/2015

3 Enjeux et contexte

3.1 La mission

Un test de régression est une vérification qui détecte si le fonctionnement d'un programme informatique a été altéré lors d'une mise à jour. L'objectif est d'éviter l'apparition de nouveaux bogues.

E-Doceo a mis en place plusieurs tests afin de s'assurer du bon fonctionnement de leurs logiciels. Ces tests sont automatisés : à chaque nouvelle modification du code, les tests vérifient le programme. Avant le début de mon stage, E-Doceo jugeait les tests trop laborieux à rédiger. E-Doceo souhaitait donc trouver des solutions afin de faciliter la rédaction des tests.

Mon stage a donc trois principales missions :
— Préconisation pour l'amélioration des procédures de tests de régression.
— Définition et mis en œuvre de bonnes pratiques.
— Implémentation des tests de régression automatisés.

L'objectif final est d'apporter de nouvelles méthodologies pour l'écriture de test de régression.

3.2 Environnement de travail

Mon stage d'une durée de trois mois s'est déroulé au siège d'E-Doceo situé à La Chapelle-Sur-Erdre, près de Nantes. J'ai intégré l'équipe de développeurs, avec laquelle j'ai travaillé tout au long de mon stage afin de proposer une solution répondant à leurs attentes.

Les logiciels d'E-Doceo sur lesquels j'ai été amené à travailler sont accessible à partir d'un navigateur web, j'ai donc principalement manipulé des technologies webs au cours de mon stage, et plus particulièrement le langage PHP.

3.3 Les outils de développement

Afin de travailler en équipe, E-Doceo a recours à certains outils. Il était donc important que les solutions proposées au cours de mon stage prennent en compte le fonctionnement, les avantages et les inconvénients de ces outils afin de proposer une solution adaptée.

3.3.1 Subversion

Apache Subversion est un logiciel de gestion de version, c'est donc un outil permettant de stocker des fichiers en harmonisant les différentes modifications apportées. Plus précisément, Subversion fonctionne de la manière suivante : un serveur stocke les différents fichiers, c'est le dépôt. Lorsque l'on souhaite modifier les fichiers, on importe localement les données, on les modifie, puis on les soumet au dépôt. Alors, Subversion met à jour les fichiers, tout en conservant un historique des modifications et des copies des anciennes versions du fichier. Lorsque deux personnes modifient le même fichier, Subversion signale alors un conflit, qui doit être résolu manuellement par un utilisateur afin de déterminer la solution à privilégier.

Concrètement, cela permet à plusieurs développeurs de travailler sur le même projet en même temps sans gêner le reste de l'équipe. Un conflit a rarement lieu et le code est continuellement mis à jour et à la disposition de tous. De plus, Subversion possède de nombreux outils permettant de faciliter la manipulation des différentes versions, l'historique des versions permettant de facilement annuler une modification et de conserver une trace des révisions. Enfin, Subversion ne fait pas réellement une copie des fichiers pour chaque version, en réalité il sauvegarde uniquement les modifications effectuées. Cela permet de posséder une multitude de versions du même projet, tout en prenant peu de place sur le serveur. On peut ainsi avoir des fichiers communs à toutes les versions, appelés le tronc, et des fichiers modifiés en fonction d'un client par exemple, appelés les branches.

En finalité, un logiciel de gestion de version est un outil indispensable pour toute entreprise intégrant une équipe constituée de nombreux développeurs, et Subversion est une solution populaire et ayant fait ses preuves, même si certains lui préfère Github, plus moderne.

3.3.2 TortoiseSVN

TortoiseSVN est un plugin de Windows permettant de donner une interface graphique à Subversion. Il permet de différencier les fichiers modifiés des fichiers à jour, et de facilement soumettre des fichiers, ainsi que de les importer, les mettre à jour, ou créer des branches, gérer un conflit, etc. TortoiseSVN facilite l'utilisation de Subversion, la gestion de versions devenant un jeu d'enfant.

3.3.3 Jenkins

Jenkins est un outil d'intégration continue, c'est-à-dire qu'il a pour but de vérifier que les modifications apportées au projet ne génèrent pas une régression. Jenkins se présente sous la forme d'un tableau de bord, regroupant les différents dépôts des projets de l'entreprise. À chaque projet, on associe une batterie de tests qui sera effectuée à chaque fois qu'une modification est apportée au projet. Et Jenkins sert simplement à automatiser ces tests, tout en donnant des informations utiles à l'aide de plugin, comme la couverture des tests ou la qualité du code. Le principal atout de Jenkins est de posséder de nombreux paramètres afin de s'adapter à n'importer quelle configuration.

Jenkins est donc principalement utilisé en tant que plateforme permettant d'avoir une vue d'ensemble des projets, tout en visualisant l'avancement des tests en cours et ceux à venir.

En conclusion, Subversion, TortoiseSVN et Jenkins sont tous les trois des outils employés par E-Doceo que j'ai été amené à utiliser au quotidien. Cela m'a poussé à adapter mes solutions afin qu'elles fonctionnent en harmonie avec ces logiciels, les tests devant être compatibles avec Jenkins. La question est donc de choisir les outils à favoriser afin de créer nos tests.

3.4 Les frameworks de test

En informatique, il est nécessaire de tester son code afin d'éviter l'apparition de bogues. Il existe des outils permettant de construire ces tests, en fonction du langage utilisé et du type de test que l'on souhaite effectuer. Cette partie s'attarde sur les différents frameworks dont E-Doceo a recours, en précisant les avantages procurés par ces tests.

3.4.1 PHPUnit

PHPUnit est le framework de référence pour effectuer des tests unitaires en PHP. Un test unitaire vérifie une portion du programme (une unité), indépendamment du reste du code. Ces tests sont construits de la manière suivante : on spécifie des valeurs d'entrées d'une fonction, et on indique la valeur attendue en sortie. Si la valeur de sortie correspond, alors le test réussi, sinon il échoue. Le but étant de créer une batterie de tests couvrants tous les cas qui nous intéresse. L'idéal serait qu'un test soit associé à chaque fonction, et que chaque test vérifie tous les cas possibles, mais cela est humainement impossible, le

nombre de tests à rédiger étant colossale. On est donc amené à ne pas tester les fonctions et les cas les plus triviaux, et à privilégier les plus critiques.

PHPUnit permet donc de tester les différentes fonctions des programmes PHP d'E-Doceo. Avec PHPUnit, à chaque classe du programme est associé une classe dite TestCase, qui contient les tests associés aux fonctions du programme. Ainsi, la structure des tests a alors une structure similaire aux classes testées. Un autre des atouts de PHPUnit est d'apporter des outils supplémentaires afin de créer des tests efficaces, comme les annotations, les bouchons, etc. PHPUnit est donc un framework à la fois simple et complet, ce qui justifie sa notoriété dans le domaine des tests unitaires.

3.4.2 Selenium

Selenium est une extension de PHPUnit permettant de créer des tests fonctionnels, c'est-à-dire des tests vérifiant les fonctionnalités de notre logiciel. La principale différence avec les tests unitaires est que les tests sont effectués en boîte noire, le but étant de tester le programme avec le même point de vue que l'utilisateur final. Cette approche a pour but de vérifier si les principales fonctionnalités du logiciel sont opérationnelles, indiquant que notre programme fonctionne normalement ou non. Cette approche permet de détecter les bogues auxquels les utilisateurs peuvent être confrontés.

Selenium est donc un framework testant les fonctionnalités d'une page web qui marche de la manière suivante : il lance un navigateur web, il accède à l'URL désiré puis il éxecute un scénario prédéfini tout en vérifiant si l'on obtient le résultat attendu. Les technologies web étant nombreuses et parfois contraignantes, Selenium a choisi de tester directement le site à partir d'un navigateur web dans le but d'assurer d'éviter tout problème de compatibilité. De plus, on peut effectuer plusieurs fois les tests en changeant de navigateur web afin de tester les différents rendus possibles entre Mozilla Firefox ou Internet Explorer par exemple. Mais passer par un navigateur ralenti également les tests, le chargement des pages étant pris en compte. Les tests deviennent alors plus laborieux à construire, puis qu'il est nécessaire d'attendre plusieurs minutes avant de savoir si le notre test est opérationnel.

Afin de cliquer sur un bouton par exemple, Selenium doit localiser l'élement sur la page HTML et pour cela, plusieurs type de localisateurs sont utilisables, chacun avec ses avatanges et ses inconvénients :

— L'ID de l'élément qui est unique par page et qui est rapide d'accès. Mais tous les éléments n'ont pas forcement une ID.

— Le nom de l'élément, auquel on peut ajouter la valeur ou l'index de l'élément. Cela s'avère particulièrement utile pour les listes. Mais tous les éléments n'ont pas forcement un nom, et un nom n'est pas unique. De plus, sur les logiciels d'E-Doceo, le changement de langue peut modifier ce paramètre.
— Si l'élement que l'on souhaite localiser est un lien vers une autre page, on peut localiser l'élément avec le texte du lien. Le désavantage étant que le texte du lien change lors de la traduction du site dans une autre langue. Ce type de localisateur est donc à éviter.
— Le selecteur CSS de l'élément, qui est plus compliqué à utiliser mais qui permet d'accéder aux élément ne possédant ni nom ni ID en utilisant la structure des balises HTML. C'est le type de localisateur le plus utilisé car il offre beaucoup de possibilité, il est rapide et il n'est pas difficile à utiliser dès lors que l'on a l'habitude du langage CSS.
— Enfin, on peut utiliser le langage XPath qui rend accessible tous les éléments de la page quel que soit la façon dont on souhaite y accéder. Mais l'utilisation de XPath est plus compliqué et plus lente que toutes les solutions précedentes. On aura donc tendance à utiliser XPath en dernier recours, pour des éléments souvent dynamique et donc difficile à identifier.

Selenium est donc un outil offrant de nombreuses possibilités, mais qui peut poser des limites techniques dès lors que les tests s'avèrent trop long à effectuer.

4 Réflexions et réponses

4.1 Observations

4.1.1 Analyse de l'existant

Dans un premier temps, j'ai analysé les tests existants afin de déterminer les améliorations possibles et lesquels sont à appliqués en priorité. L'équipe d'E-Doceo s'étaient principalement concentré sur deux types de tests : les test unitaires et fonctionnels.

Les test unitaires permettent de tester si une fonction produit le résultat attendu. Ces tests doivent être réalisés dès lors que la conception d'une fonction s'est avèré complexe, permettant ainsi de s'assurer qu'aucun effets indésirables ne survient. Un test est souvent long à écrire, et le nombre de fonctions à couvrir par des tests s'avère faramineux en prenant en compte l'ensemble des logiciels d'E-Doceo. De plus, il est difficile de déterminer les cas pertinents à tester en vue des nombreux paramètres à prendre en compte. Ces deux principaux problèmes m'ont amenés, après en avoir discuté avec le reste de l'équipe, a préféré les tests fonctionnels aux test unitaire, la priorité étant le bon fonctionnement du logiciel. Il est tout de même nécessaire de continuer d'utiliser des tests unitaires lorsque qu'une fonction s'avère compliqué ou souvent employée, afin de compléter la base de tests déjà écrit dans le but d'assurer le bon déroulement des principales fonctions.

Les test fonctionnels donc, sont réalisés dès lors que l'équipe d'E-Doceo juge qu'une fonctionnalité doit être associé à un test de régression, afin d'assurer que cette dernière soit utilisable par les clients. Lors de la réception de tickets signalant qu'une fonctionnalité n'est plus en état de marche, un développeur est assigné à la correction de ce bogue et, si il pense cela nécessaire, il peut rédiger un test afin d'éviter une nouvelle régression. La encore, le nombre de fonctionnalités des logiciels d'E-Doceo étant trop élevé, il est impossible de toutes les tester. Mais associer à chaque correction de bogue un test de régression peut, en hypothèse, permettre d'assurer les fonctionnalités qui sont principalement utilisés par les clients d'E-Doceo.

Ces tests sont rédigés à l'aide l'API de Selenium, et ils utilisent directement les fonctions fournit par l'API. Il existe deux principaux types de fonctions : celle qui permettent de naviguer sur des pages webs et celles qui vérifient un élément de la page. Les premières servent à mettre en place le contexte du test, c'est-à-dire les conditions dans lesquels on souhaite tester si notre fonctionnalité marche ou non, tandis que les secondes sont les tests en eux-mêmes. Dans les faits, les tests d'E-Doceo sont majoritairement composés de fonctions permettant de mettre en place le contexte car les

fonctionnalités à tester sont souvent complexes. De ce constat, j'en est déduis que le principal objectif était de réduire le temps nécessaire à la mise en place du contexte. En analysant le code, un second constat est également apparu : les tests étaient difficilement lisible à cause des localisateurs. En effet, ceux-ci se basant sur les balises HTML de la page, il n'est pas rare de localiser un élément avec un attribut ambigu. Par exemple, si l'on souhaite localiser un bouton avec son ID qui vaux "bt2", alors la ligne de code correspondant au clique s'écrira de la manière suivante : "Je clique sur l'ID bt2". Le test devient alors rapidement difficile à comprendre sans le code HTML de la page. Enfin, j'ai également constaté de nombreuses redondances dans le code dû aux nombreuses pages parcourues lors des tests.

J'ai alors pensé à intégrer le design pattern State afin de rendre le code plus lisible et moins redondant. Le DP State consiste a donner un état a un objet. Lorsque l'objet change d'état, alors les fonctions de l'objet changent. Ici, l'objet en question serait la page actuellement parcourue, les états seraient les différentes pages webs de notre logiciel et les fonctions les différentes fonctionnalités de la page. La figure ci-dessous illustre le principe du DP, la classe Context étant notre classe de test, et les classes States étant les pages web. J'avais donc une première solution permettant de rendre le code plus intelligent et structuré.

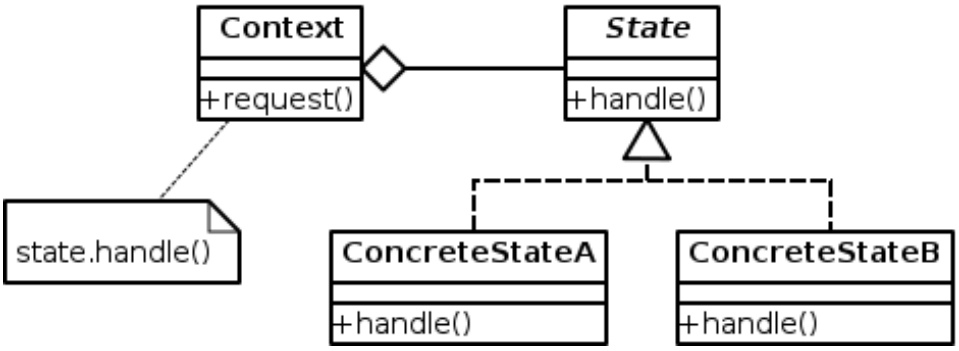

FIGURE 3 – Design pattern State

En conclusion, les difficultés liés aux tests unitaires, l'efficacité des tests fonctionnels et les défauts que j'ai constaté à propos de l'API de Selenium m'ont amenés à me concentrer en priorité sur les tests fonctionnels et à rechercher des solutions pour rendre l'utilisation de Selenium plus aisé.

4.1.2 Premières réponses

La première solution permettant de simplifier la rédaction des tests fonctionnels est l'utilisation de Selenium-IDE, un plugin de navigateur web per-

mettant de construire des tests en parcourant simplement le site. Cette solution permet en théorie de rapidement réaliser un scénario de test, et c'est pourquoi l'équipe d'E-Doceo l'a utilisé dans le passé, mais le plugin s'avère perfectible et limité. En effet, dès lors qu'un scénario est long, Senelium-IDE peut-être amené à effectuer une erreur lors de l'exécution du test. De plus, le plugin ne permet pas d'intégrer du code PHP, ce qui limite l'écriture des tests. Il est possible d'exporter les tests en PHP, mais le code est alors illisible et nécessite d'être réécrit et commenté. Toutes ces raisons rendent Selenium-IDE impossible à utiliser afin de construire un scenario, mais le plugin peut s'avérer pratique en tant qu'outils pour aider à la conception du test.

Une deuxième solution mise en avant par Selenium est l'utilisation du design pattern PageObjects. Ce design pattern consiste à réaliser une classe par page, une classe regroupant les fonctionnalités de la page. Ce DP se rapproche donc fortement de mon idée initial d'intégrer un DP State. Ce DP étant indiqué comme une bonne pratique par Selenium, j'ai donc essayé de rapidement le mettre en pratique avec notre cas.

4.2 Le design pattern PageObjects

4.2.1 Principe et avantages

Le design pattern PageObjects est une astuce de programmation née avec l'apparition des tests fonctionnels sur PHP, et notamment avec la popularité grandissante de Selenium. Ce DP n'est en effet qu'utile dans la conception de tests fonctionnels en PHP, puisque les logiciels construits avec la notion de page sont principalement sous la forme de site internet.

Le principe du DP est simple : à chaque page du logiciel correspond une classe et à chaque fonctionnalité de la page correspond une fonction. Par exemple, si la page d'accueil contient un bouton, alors il doit exister une classe contenant une fonction qui clique sur ce bouton. Alors, lorsque dans un test nous souhaitons cliquer sur le bouton, nous appelons cette fonction. Cette approche permet de séparer notre test et la structure des pages HTML. Le principal objectif est de faciliter la navigation sur les pages. En effet, cette séparation a plusieurs avantages :

— Les fonctionnalités déjà utilisées dans un autre test ont déjà leur fonction. Il n'est donc plus nécessaire de rechercher un localisateur pour l'élément, et cela permet donc de construire une bibliothèque de fonctions grandissantes au fur et à mesure, rendant les tests de plus en plus rapides à écrire.

— Le code est plus facile à maintenir : si une page de notre logiciel est modifié, alors il suffit d'adapter la classe de la page. Il n'est donc plus nécessaire de changer la totalité des tests parcourant cette page.

— Les tests sont également plus lisibles et intuitifs, les noms des fonctions des classes PageObjects décrivant la fonctionnalité utilisée.

— Enfin, les tests ne comportent plus l'aspect technique des pages HTML, c'est maintenant les classes PageObjects qui attendent par exemple le chargement de la page.

Ces nombreux avantages m'ont donc amené à implémenter le design pattern dans les tests que j'ai réalisés pour E-Doceo.

4.2.2 Implémentation

Concrètement, le design pattern PageObjects se présente sous la forme illustrée par la figure 4 ci-dessous :

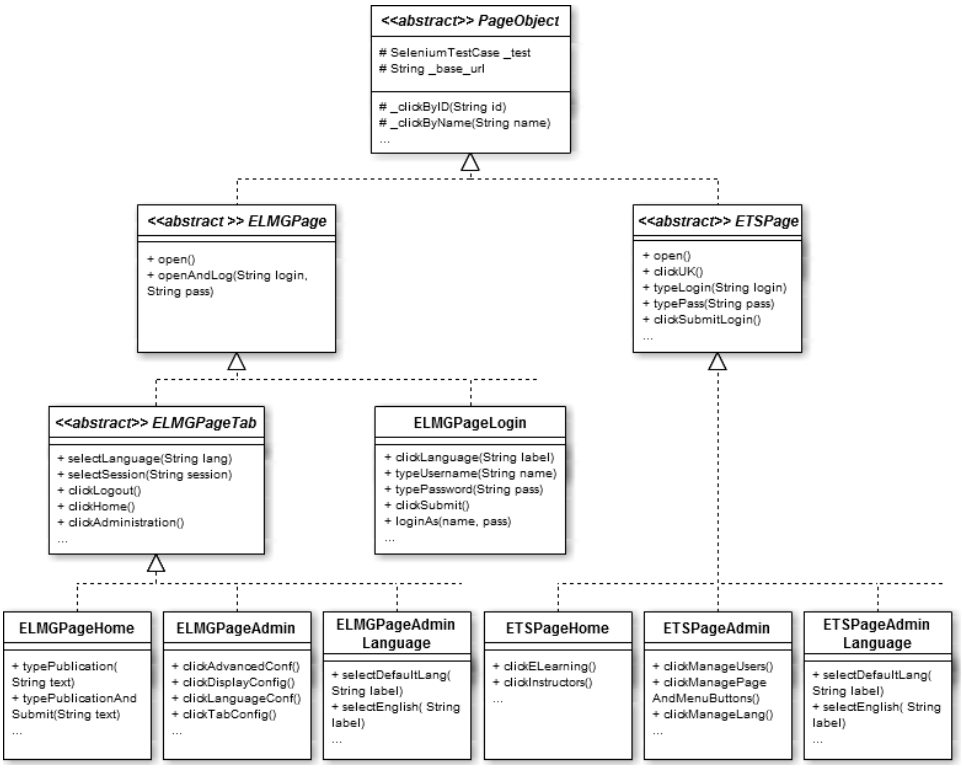

FIGURE 4 – Design pattern PageObjects

La classe PageObject est commune à tous les logiciels, et c'est cette classe qui utilise l'API Selenium. Le premier but de cette classe est de regrouper

tous les attributs communs aux pages, comme le test qui parcourt la page, le constructeur, ou encore l'URL de base des pages, qui sert à construire des chemins relatifs. Le second but de cette classe est de définir des fonctions protégées servant à faciliter l'écriture des tests, et surtout à être moins dépendant de l'API Selenium. Le but étant de ne pas utiliser l'API dans nos classes Page Object, mais ces fonctions. Ceci dans le but de faciliter le changement d'API si cela s'avère nécessaire dans le futur, ou si l'API Selenium subit une mise à jour importante. Il était important que cette classe soit claire et intuitive, afin d'intégrer plus facilement ma solution au reste de l'équipe. J'ai donc utilisé PHPDoc, un outil générant une documentation pour du code PHP, afin que l'équipe dispose de documents auxquels se référer dans le futur. Cette documentation étant réalisé à l'aide de commentaires, cela permet également de rendre le code plus lisible. Enfin, l'API Selenium n'étant pas toujours bien documenté et en anglais, notre documentation peut comporter des informations plus spécifiques à nos logiciels.

Les classes héritant de PageObject sont des classes spécifiques à un logiciel : ELMGPage pour les pages d'ELMG et ETSPage pour les pages de catalogue. Ces classes comportent donc les fonctions communes à toutes les pages, en plus d'intégrer les fonctions permettant d'ouvrir une nouvelle page dans le navigateur. La classe ELMGPageTab contient quant à elle des fonctions permettant de cliquer sur l'onglet d'ELMG, onglet commun à de nombreuses pages d'ELMG. Catalogue a un onglet similaire mais contrairement à ELMG, toutes les pages d'ETS ont cet onglet, j'ai donc mis ces fonctions directement dans la classe ETSPage.

Enfin, les classes concrètes sont les classes contenant les fonctionnalités spécifiques à la page, et chaque classe concrète doit correspondre à une unique page du logiciel. C'est donc ces pages qui vont être manipulés dans les tests.

4.2.3 Les choix et améliorations possibles

Cette partie présente les différents choix que j'ai émis, et les différentes améliorations possibles dans le futur.

Lors de la mise en oeuvre du DP, il était nécessaire de prendre une décision vis-à-vis de la transition des pages. En effet, comment gérer le changement de page, dans le cas où une fonctionnalité peut emmener à deux pages différentes. Par exemple, sur la page de connexion d'ELMG, le bouton validé peut ouvrir la page d'accueil si la connexion a réussi, mais le site peut également demander un nouvel essai si la connexion a échoué. Dans le design pattern Page Objects émit par Selenium, toutes les fonctions retournent une nouvelle page, ce qui signifie que c'est lors de l'écriture du test que l'on

doit changer de page. C'est donc au rédacteur du test de savoir quel page va être ouvert, et il existe plusieurs fonctions pour chaque fonctionnalité selon le résultat retourné, comme "clique sur le bouton valider et connexion réussi" et "clique sur le bouton validé et connexion échoué". La deuxième solution était de plus se rapprocher du DP State en laissant les transitions de pages transparentes lors de la rédaction du test. C'est donc au rédacteur des classes Page Objects de savoir quand une fonctionnalité peut changer la page courante. La méthode utilisée ici consiste à vérifier l'URL de la page après un clique sur le bouton validé : si l'URL correspond à la page d'accueil, alors on effectue la transition, sinon on ne change pas la page. Cela permet de conserver l'idée d'une fonction par fonctionnalité, et de garder les aspects techniques des pages dans les classes Page Objects.

Une amélioration possible serait de s'inspirer de l'approche "tell, don't ask" qui consiste à éviter d'interroger des données (ici les pages) pour ensuite agir en conséquence (ici les tests) au lieu de tout mettre en commun. Dans notre cas, cela signifierait que nous intégrions des tests dans le DP Page Object. Le principe "tell, don't ask" s'oppose au principe "single responsibility", qui lui, favorise la séparation des différentes taches pour que les classes s'en tiennent à un rôle précis. Dans un premier temps, on peut être amené à penser que l'objectif principal du DP Page OBject était de séparer les tests des pages, l'approche "tell, don't ask" n'a pas d'utilité ici. Mais par exemple, vérifier à chaque ouverture d'une page si l'URL du site correspond bien à la page permettrait de plus facilement d'identifier les raisons de l'échec d'un test (avec l'exemple, si un test échoue, on saurait rapidement si on est sur la bonne page ou non). De plus, l'API Selenium comprend des fonctions vérifiant un élément sans arrêter le scénario de test (contrairement à une fonction de test classique), ce qui permet d'intégrer ces vérifications sans influencer le scénario. Ces vérifications supplémentaires n'ont pas pour objectif de tester une fonctionnalité, mais d'aider à la résolution des bogues. Leur rédaction est donc optionnelle, mais peut rendre nos tests plus efficaces.

Enfin, E-Doceo souhaite migrer ses logiciels vers le framework Symfony. Ce framework possède un outil nommé Conteneur de Services, qui permet de regrouper des objets instanciés. L'utilisation de cet outil pourrait être utilisée afin d'instancier une unique fois toutes les classes Page Objects, au lieu de créer une nouvelle instance à chaque transition de page.

4.3 Le Behavior Driven Development

Le Behavior Driven Development est une méthode agile consistant à décrire les tests fonctionnels dans un langage naturel avant de réaliser les fonctionnalités. Cette méthode permet de décrire le résultat attendu par la

fonctionnalité, les tests servant alors de cahier des charges, tout en facilitant l'écriture des tests. Malgré un potentiel gain de temps et une méthode efficace afin de décrire un test, cette approche moderne des tests est rarement utilisée.

4.3.1 Behat

Behat est un framework de BDD dédié au langage PHP. Le principe de Behat est donc de décrire dans un langage naturel les tests à réaliser. Cette description doit respecter la syntaxe de Gherkin, le parseur de Behat qui définit le langage naturel utilisé. Ce parseur impose certains mots-clefs afin de décrire notre futur test fonctionnel :

— "Feature" indique que l'on test une nouvelle fonctionnalité

— "Scenario" indique que l'on va décrire un nouveau scénario de test

— "Given" indique que le contexte initial. Dans notre cas, ça sera par exemple la page ouverte au début du scénario.

— "When" indique les actions que l'on effectue afin de mettre en place le test.

— "Then" indique quand à lui, les vérifications effectuées.

Ce sont les principaux mots-clefs de Gherkin. Leur utilisation a pour but de rendre les tests plus faciles à lire, et de décrire dans un premier temps les tests de la manière la plus générique possible. Dans un second temps, une fois la fonctionnalité disponible, les tests fonctionnels seront plus faciles à rédiger. Les tests prennent alors la forme suivante :

```
Feature: La description de la fonctionnalite
    Par exemple, je vais cliquer sur le bouton administration
    et verifier si je suis bien sur la bonne page

Scenario: Clique sur le bouton administration
    Given Je suis sur la page d accueil
    And Je suis connecte en tant qu administrateur
    When Je clique sur le bouton Administration
    Then Je devrais voir la page Administration

Scenario: Un autre scenario
    ...
```

Une fois notre test décrit, chaque ligne va générer une fonction. Par exemple, la première ligne va générer une fonction "jeSuisSurLaPageDAccueil". L'objectif maintenant est de décrire le code associé à cette fonction.

4.3.2 La notion de contexte de Behat

Les fonctions générées automatiquement par Behat sont placés dans un fichier appelé le contexte. C'est donc le contexte qui regroupe toutes les fonctions de notre test.

Dans notre cas, le contexte s'avère être la page que l'on parcourt actuellement. Un problème se pose alors : il est impossible au cours d'un scénario écrit en Gherkins de signaler un changement de contexte. Cela signifie donc qu'il est impossible de changer de simplement de page au cours d'un scénario. Une extension intégrée le DP Page Objects existe, mais elle demande du temps à être mis en place. En effet, la solution consiste à donner un attribut page à la classe contexte, et de faire le lien entre les fonctions de la page actuellement parcourue avec les fonctions du contexte. On a donc une couche supplémentaire entre les tests et les pages HTML, qui n'est pas réellement cohérente, ni intuitive, et qui n'apporte rien au code.

Ce problème s'explique par le fait que dans un logiciel simple, il n'est pas nécessaire de parcourir plusieurs pages afin de tester une unique fonctionnalité. Mais les nombreuses possibilités offertes par les logiciels d'E-Doceo requièrent des tests compliqués et demandants bien souvent de parcourir plusieurs pages.

En principe, il serait possible de recourir à Behat pour quelques tests, mais les limites du framework peuvent rapidement devenir gênantes. De plus, le principal atout de Behat est de rendre les codes plus lisibles. Mais le DP Page Objects permet déjà de clarifier les tests. Enfin, Behat est avant tout un outil de BDD. Or, les fonctionnalités à tester sont d'ores et déjà conçues. L'utilisation de Behat ici perd donc son sens premier. J'en ai donc conclut que Behat n'était pas une solution adéquate pour les logiciels de la taille et de la complexité de ceux d'E-Doceo, mais il s'avère être un atout au commencement d'un nouveau projet afin d'orienter son développement.

4.4 Standardisation des tests fonctionnels

Behat n'étant pas une solution adéquate, il est nécessaire de définir une nouvelle méthodologie afin de structurer les tests. Définir de nouvelles règles permet de rendre les tests plus homogène, facile à comprendre et modulaire. Cette partie va donc présenter les nouveaux standards à respecter des tests fonctionnels d'E-Doceo.

4.4.1 Définition des bonnes pratiques

Il est tout d'abord nécessaire de définir les bonnes pratiques à adopter lors de l'écriture d'un test :

— Lors de la rédaction des classes PageObjects, il faut favoriser les localisateurs ID, CSS et par nom d'élément. Ces localisateurs sont rapides et permettent de localiser la plupart des éléments d'une page HTML.

— Il est également important d'éviter d'utiliser des localisateurs basés sur des éléments qui seront amenés à être modifié, comme un nom d'image ou le texte de l'élément qui change en fonction de la langue.

— Il ne faut jamais utiliser l'API Selenium directement dans un test ou dans une classe héritant de PageObject. Rédiger des fonctions protégées dans la classe PageObject redéfinissant l'interface de Selenium permet d'être moins dépendant de l'API.

— Il est favorable de rédiger le code le plus modulaire possible : si un élément est commun à plusieurs pages, alors il faut écrire une classe spécifique à cet élément, et l'ajouter en tant qu'attribut aux classes PageObjects intégrant cet élément.

— Il est important de maintenir à jour la documentation de la classe PageObject, cette dernière regroupant des fonctions communes à de nombreux tests et étant souvent réutilisé.

— Il est également important de commenter les tests, en décrivant leurs objectifs, afin de faciliter leur lecture.

— Il peut être moins important de documenter les classes PageObjects, le nom des fonctions et des classes décrivant normalement leur rôle. De plus, pour la rédaction d'un test fonctionnel, on peut être amené à rédiger de nombreuses classes et fonctions. Les commenter peut donc s'avérer long et peu utile.

4.4.2 Structure des fichiers de tests

E-Doceo utilise Subversion, il est donc nécessaire de définir où les différents fichiers sont situés. E-Doceo dispose d'un dépôt commun à tous les projets. C'est donc dans ce dépôt les classes communes à tous les tests sont déposés. Les classes du DP PageObjects sont également disponibles sur ce dépôt, certains tests pouvant parcourir plusieurs logiciels au cours d'un même scénario. Quant aux tests, la configuration de Jenkins requiert de regrouper les tests dans un dossier spécifique à chaque logiciel. Les tests sont donc spécifiques à un logiciel.

4.4.3 Les classes de test

Il est maintenant important de définir l'architecture des classes de tests, dans le but de les rendre plus modulaire. L'objectif est de faciliter la prise en compte du travail déjà réalisé, afin de le réutiliser et donc rendre la conception des tests plus rapide. La nouvelle architecture des tests prend la forme suivante :

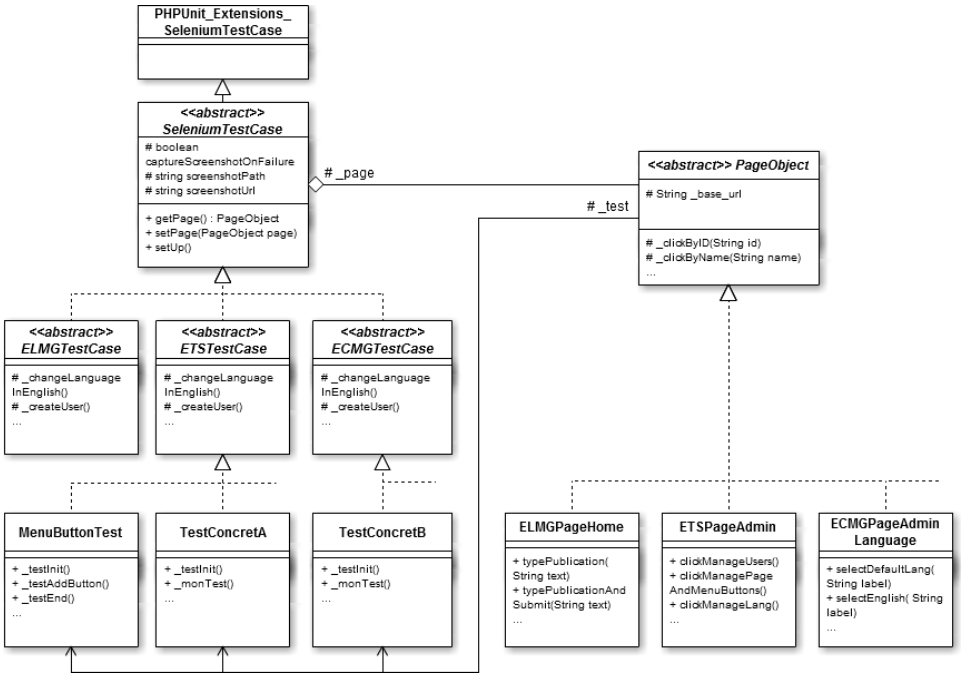

FIGURE 5 – Diagramme UML des tests fonctionnels

La classe SeleniumTestCase regroupe les attributs et fonctions communs à tous les tests. Elle hérite de PHPUnit_Extensions_SeleniumTestCase qui est la classe permettant d'utiliser l'API de Selenium, et qui possède plusieurs options à configurer, comme le navigateur web utilisé ou la possibilité de prendre une capture d'écran lors de l'échec d'un test. La classe SeleniumTestCase permet donc de paramétrer nos tests, et également de mettre en place le DP PageObject, en ajoutant l'attribut "page" avec ses accesseurs à tous les tests. Enfin, SeleniumTestCase regroupe tous les attributs et fonctions que l'on souhaite retrouver dans tous les tests, comme par exemple l'URL de base du site. Isoler les fonctionnalités de Selenium a pour objectif de faciliter un éventuel changement d'API, les deux principales classes à modifier étant SeleniumTestCase et PageObject si nous ne souhaitons plus utiliser Selenium.

Les classes ELMGTestCase, ETSTestCase et ECMGTestCase servent à re-

grouper les attributs et fonctions communes à tous les tests de leurs logiciels respectifs. Par exemple, leur constructeur initialise l'attribut "base_url" qui est l'URL de base du logiciel, ou les fonctions "changeLanguageIn" qui change la langue du site. Les classes de tests ont alors des fonctions permettant de facilement mettre en place un contexte, comme "createUser" qui crée un utilisateur. Il reste donc à définir la forme finale des tests, afin de les rendre plus lisible et intuitif.

4.4.4 Granularité des tests

La classe "MenuButtonsTest" de la figure 5 est un test fonctionnel réalisé au cours de mon stage intégrant le DP PageObjects. Afin de faciliter la conception des tests, certains paramètres du site sont initialisés pour chaque test, comme de mettre le site en anglais. La fonction "testInit" initialise donc notre test. Afin que Selenium lance une fonction, il est nécessaire qu'elle soit en visibilité publique et comporte le mot "test", c'est pour cela que chaque classe de test possède cette fonction. Ensuite, la fonction "testAddButton" exécute notre test. Il commence par ouvrir la page d'accueil d'ETS, puis il se connecte en tant qu'administrateur, ensuite il accède à la page d'ETS qui contiennent la fonctionnalité à tester, et enfin nous testons notre fonctionnalité.

Cette approche, similaire à celle utilisée jusqu'alors chez E-Doceo, possède plusieurs désavantages. Initialiser chaque classe de test avec des paramètres identiques allonge le temps d'exécution des tests, et cela s'avère utile uniquement si un test modifie un paramètre concerné. Il serait plus avantageux d'initialiser une unique fois ETS, puis de lancer les tests tout en s'assurant qu'ils ne modifient pas le contexte. Une seconde optimisation possible est de faciliter l'accès à la page qui contient la fonctionnalité à tester, en créant une fonction qui ouvre directement cette page en indiquant l'URL. Naviguer sur le site prend du temps, et ce n'est pas les pages précédentes que nous souhaitons tester. L'idée est finalement de se rapprocher de la mise en forme des tests de Behat : on donne dans un premier lieu le contexte du test et la page à ouvrir, puis on utilise les fonctionnalités et enfin on vérifie le résultat. Chaque scénario de test aura alors pour but de vérifier quelque chose de précis, et les classes de tests regrouperont les tests effectués sur une fonctionnalité donnée.

Cette nouvelle approche permet de plus facilement retrouver les tests déjà réalisée afin de les compléter par exemple. La suite et fin de mon stage va donc consister à mettre en place cette nouvelle méthodologie au sein d'E-Doceo.

5 Conclusion

En conclusion, intégrer une équipe de développeurs au sein d'un éditeur de logiciels m'a permis de découvrir les réalités des métiers informatiques et des projets professionnels, en travaillant pour E-Doceo mais également en observant l'équipe qui m'a accompagné. J'ai également appris à professionnaliser mon travail, en l'exposant à mes collègues et en écoutant leurs attentes. J'ai aussi découvert une nouvelle forme d'apprentissage, le digital learning, ainsi que les différents acteurs, enjeux et contraintes de ce domaine. En outre, travailler en entreprise m'a amené à utiliser des outils que je serais sûrement amené à réutiliser tel que Subversion, Jenkins ou encore Selenium. Enfin, concevoir des tests de régressions m'a beaucoup appris sur le domaine des tests, aussi bien unitaires (mocks, mutation testing...) que fonctionnel, et également sur les méthodes agiles, comme le Behavior Driven Development.

L'entreprise E-Doceo qui m'a accueilli pendant ce stage avait besoin de redéfinir les méthodes de conception des tests dans le but de les rendre plus rapides à réaliser, plus lisible et plus efficace. Après plusieurs observations, nous avons alors décidé de nous concentrer sur la mise en place de tests fonctionnels car jugé plus efficace. Puis, la mise en place du design pattern PageObjects et la standardisation des tests a permis d'apporter une réponse à ces problématiques. Les tests seront de plus en plus rapides à concevoir, la liste des fonctions réutilisables s'enrichissant à chaque nouveau test. La nouvelle structure des tests, la documentation et la définition des bonnes pratiques rendront les tests plus faciles à comprendre, à compléter et à maintenir pour l'ensemble de l'équipe. Enfin, la conception d'un test à chaque nouvelle correction de bogue jugé important peut potentiellement réduire leur nombre de manière efficace.

La suite et fin de mon stage va maintenant consister a mettre en place les solutions mise en avant dans ce rapport, et donc à écouter les remarques et suggestions du reste de l'équipe de développeurs. Je vais également être amené à écrire de nouveaux tests, et peut-être à réécrire les tests déjà existants, afin de les standardiser, de rendre leur exécution plus rapide et d'enrichir les classes de tests de fonctions réutilisable par la suite. Et une fois ma mission terminé, pourquoi ne pas recourir aux méthodes agiles comme le BDD, en concevant les tests avant les fonctionnalités ?

6 Glossaire

API Ensemble de classe et de fonctions servant à utiliser un logiciel.

Behavior Driven Development Méthode agile qui consiste dans un d'abord écrire les tests fonctionnels dans un langage naturel, pour ensuite créer les fonctionnalités correspondantes.

Classe Notion de POO. Ensemble de fonctions et d'attributs définissant un objet.

Classe virtuelle Formation dispensée à distance à l'aide d'internet.

CSS Langage informatique décrivant la présentation d'un document HTML.

DP Design Pattern. Normes de POO dont le but est de structurer le code de manière astucieuse.

Digital learning Désigne l'utilisation de nouvelles technologies dans du contenu de formation.

E-Learning Apprentissage à distance au moyen de nouvelles technologies.

ECMG LCMS d'E-Doceo.

ELMG LMS d'E-Doceo.

ETS Logiciel complémentaire à ELMG, qui est un catalogue de parcours de formation destiné aux apprenants.

Formation en présentiel Désigne une formation où les apprenants et les formateurs sont réunis en une même salle.

Framework Ensemble d'outils destiné à faciliter le travail d'un développeur.

HTML Langage informatique décrivant les données d'une page web.

LCMS Logiciel web destiné aux formateurs afin de concevoir un parcours de formation.

LMS Logiciel web destiné aux apprenants afin de suivre un parcours de formation.

Méthode agile Les méthodes agiles sont des pratiques rendant la gestion de projet plus pragmatique, dans le but de satisfaire au mieu le client.

Parseur Analyseur syntaxique qui, à l'aide de mots-clefs, définit un langage.

PHP Langage de programmation destiné à programmer des logiciels utilisant internet.

POO Programmation Orientée Objet. Paradigme de programmation dont le but est de rendre plus modulaire le code en le divisant sous forme de brique de code : les objets.

Rapid-learning Méthode pédagogique qui a pour vocation un apprentissage rapide.

Serious game Jeu vidéo pédagogique.

Test de régression Test vérifiant qu'une mise à jour du logiciel n'a pas altéré son fonctionnement.

Test fonctionnel Test vérifiant une fonctionnalité du logiciel.

Test unitaire Test vérifiant une fonction d'un programme.