Synthèse

The Relationship between Accessibility and Usability of Websites

PRÉSENTATION

L'article étudié s'attache dans un premier temps à définir les notions d'accessibilité et d'utilisabilité dans les interfaces utilisateurs de sites web. Tandis que la définition de l'utilisabilité est fixée : *« the extent to which a product [or website] can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use »*, la définition de l'accessibilité reste encore floue. L'organisme du W3C estime que cette notion englobe la façon qu'on les utilisateurs déficients d'accéder et d'interagir avec les éléments du Web. On se rapproche donc plus de l'utilisabilité pour les personnes déficientes que d'une définition globale de l'accessibilité. Pour palier à ce manque, le W3C a publié (et met encore à jour) un document de recommandation (WCAG : Web Content Accessibility Guidelines). Ces directives s'attachent à mesurer les problèmes d'accessibilité qui peuvent être rencontrer dans la navigation web, et des conseils pour les résoudre.

Cette étude amène vers la relation entre l'accessibilité et l'utilisabilité, très peu explicitée. Tandis que le chercheur **Thatcher** propose que l'accessibilité est un sous-ensemble de l'utilisabilité, son compatriote **Shneiderman** avance le concept d'utilisabilité universelle. Cet article s'attache donc à définir une nouvelle relation entre ces deux entités.

Les auteurs avancent que ces deux ensembles se chevauchent, et déterminent trois catégories :

- Les problèmes rencontrés uniquement par des utilisateurs déficients sont classés comme « **pure accessibilité** ».
- Les problèmes rencontrés uniquement par des utilisateurs non-déficients sont classés comme « **pure utilisabilité** ».
- Les problèmes rencontrés par les deux panels d'utilisateurs sont classés comme « **utilisabilité universelle** ».

L'article spécifie également qu'il convient de prendre en compte l'importance que les utilisateurs accordent à un problème donné. En effet, imaginons un titre dans un site web quasiment illisible (texte blanc sur fond blanc par exemple), et qui ne respecte pas les conventions définies par le WCAG. Par conséquent, les utilisateurs déficients ne prendront pas connaissance de ce texte, ce qui classe le problème comme **majeur** voir **bloquant**. Par contre, les utilisateurs non-déficients s'orienteront vers un problème d'importance **mineure**, car il leur suffit par exemple de surligner le texte pour le rendre lisible.

Sur cette base de critères d'évaluation, les auteurs ont procédé à une étude centrée utilisateurs pour donner du poids à leur argumentation, et démontrer l'insuffisance des recommandations fournies par deux organismes majeurs : **WCAG**, énoncé précédemment, et **HHS** Guidelines (Health & Human Services). Il conviendra d'étudier les informations fournies par ces deux organismes, et de présenter l'expérience développée par les auteurs ainsi que ses résultats.

WCAG & HHS GUIDELINES

Le département américain de la Santé et des services aux personnes (HHS) propose des recommandations, et plus particulièrement un chapitre sur l'accessibilité. Les directives identifiées sont destinées aussi bien à des développeurs Web qu'à des designers. Elles déterminent des règles qu'un site internet se doit de satisfaire pour augmenter son impact avec des utilisateurs déficients. Il n'y a aucune prise en compte de la plate-forme sur laquelle l'utilisateur évolue, ni son environnement. Seul la présence de la solution sur le web compte.

Leur système de classification est basé sur deux critères :

- Importance relative (*relative importance level)* : note sur 5 donnée par 16 critiques (50% designers web, 50% spécialistes en utilisabilité). L'évaluation est basée sur la question « Quelle importance à cette recommandation dans le succès d'un site internet ? »
- La pertinence de la solution : note sur 5 donnée par 8 critiques (chercheurs, professionnels et auteurs expérimentés). Elle représente la puissance de la solution pour résoudre un problème d'accessibilité.

Ces recommandations s'attachent à présenter des solutions à prendre en compte au moment de l'exécution, pour faciliter la vie de l'utilisateur final. Par exemple, la section 12 recommande de donner un titre à chaque frame, pour permettre une meilleure navigation à un public déficient visuel. La section 5 quant à elle conseille de fournir des descriptions textuelles pour des éléments non-textes.

D'une façon similaire, le W3C a publié en 1999 la version 1.0 d'un lot de recommandations orientés sur l'accessibilité (WCAG) dans les interfaces web. Une version 2.0 est encore en cours de rédaction, c'est pour cette raison que nous ne nous intéresserons ici qu'à la version initiale. Si le contexte d'usage de ces recommandations est le même que pour HHS, le système de classification diffère. Le WCAG identifie trois priorités possibles pour chaque recommandation (*priority level)* :

- Le développeur web **doit** satisfaire cette directive. Dans le cas contraire, des utilisateurs peuvent ne pas avoir accès à l'information.
- Le développeur web **devrait** satisfaire cette directive. Sinon, des groupes de personnes vont trouver difficile d'accéder au contenu désiré.
- Le développeur web **peut** satisfaire cette directive pour améliorer l'accessibilité de sa solution et satisfaire d'autant plus les utilisateurs avec des déficiences.

Ces deux organismes ne s'attachent donc pas à définir un modèle pour construire une solution web respectant l'ensemble des points d'accessibilité énoncés, mais fourni des conseils pour augmenter la pertinence d'un site internet dans ce domaine.

ÉTUDE UTILISATEURS

L'article propose une étude centrée utilisateurs pour répondre à trois questions de recherche :

- Quelles sont les relations entre les problèmes rencontrés par les personnes voyantes et non-voyantes ?
- Si des problèmes sont identifiés dans les deux panels d'utilisateurs, est-ce que la sévérité est plus élevée chez les non-voyants ?
- Existe-t-il une relation entre les notes attribués par les utilisateurs de l'étude, les notes des experts, et les notes données par les organismes WCAG et HHS pour la sévérité des problèmes d'accessibilité/utilisabilité ?

L'étude est composée de douze participants, six voyants et six non voyants. Une première estimation leur a été demandée sur leur aisance avec les systèmes informatiques, et leur fréquence d'utilisation. Les auteurs ont décidé d'évaluer les problèmes d'utilisabilité/accessibilité sur deux sites internet : T-Mobile et Orange. Le but des participants à l'étude est d'effectuer sept tâches, idoines sur ces deux sites, avec une difficulté progressive (au niveau du nombre de clics, de la difficulté de localiser l'information, le nombre de pages à consulter…).

Les participants sont évalués sur le nombre de tâches menées à bout, le nombre de pages visitées, le nombre de problème rencontrés, avec leur sévérité associée.

Les résultats présentés insistent sur les trois points énoncés ci-dessus : **l'intersection** des problèmes entre les deux cibles, la **sévérité** associé à chacun et la **concordance** avec les recommandations des deux organismes.

Dans un premier temps, **l'intersection** calculée est moins forte que les attentes des auteurs. Autrement dit, le pourcentage de problèmes identifiés par les deux panels d'utilisateurs est plus faible que prévu. Malgré tout, les 14% de problèmes communs, pour deux catégories d'usagers très éloignés (les personnes déficientes visuelles ont utilisé un lecteur d'écran), démontrent qu'il existe bel et bien des relations entre l'utilisabilité et l'accessibilité, trop souvent négligé par les chercheurs et l'industrie du web.

Les résultats obtenus vis à vis de la **concordance** de la sévérité des problèmes entre les participants de l'étude et les organismes WCAG et HHS sont plus satisfaisants pour les auteurs. En effet, l'étude ne démontre **aucune** relation entre les notes des organismes (« *relative importance level* » pour HHS et « *priority level* » pour WCAG) et les usagers interrogés. Cette conclusion atteste que les systèmes de classification des deux organisations sont trop faibles, et n'évalue pas assez précisément l'importance des problèmes rencontrés par les utilisateurs.

Les conclusions de cet article mettent donc en avant deux problèmes en matière d'accessibilité/utilisabilité sur le web :

- L'utilisabilité et l'accessibilité sont deux concepts liés.
- Les organismes traitant d'accessibilité sur le web ne couvrent pas l'ensemble des problèmes rencontrés par des utilisateurs, déficients ou non. De plus, leurs systèmes de classification, trop faible, n'est pas à la hauteur de la réalité.

AVIS ET PERSPECTIVES

Cet article ne s'est donc pas attaché à présenter une solution, mais s'est plutôt concentré sur une identification plus fine des problèmes existants en matière d'utilisabilité et d'accessibilité sur des plate-formes web. Ces deux domaines sont très sensibles et ne sont pas du tout résolus actuellement, et effectuer un rapport de fond sur la relation qu'il peut exister entre le deux permet de voir les « contraintes » développeurs liées à l'accessibilité sous un autre angle.

S'il est dommage que l'étude centrée utilisateurs réalisée ne permette pas de donner assez d'éléments de réponse pour les trois questions soulevées par les auteurs, les deux conclusions énoncées précédemment clarifie la situation en matière d'accessibilité sur le web, et pousse d'autres travaux à se concentrer sur ces deux perspectives.

Le rapprochement de ces deux concepts, et les liaisons existantes entre eux, amène à la remarque suivante : l'interface utilisateur finale présente des problèmes communs à l'accessibilité et l'utilisabilité. Une solution pourrai être de monter en abstraction, et de faire émerger un modèle résolvant les problèmes communs aux deux notions.

En prenant comme référence le *framework* CAMELEON (http://www.w3.org/2005/Incubator/model-based-ui/XGR-mbui-20100504/#crf), on observe que les recommandations identifiées (et les notes associées) par les organismes WCAG et HHS, se situent au niveau de l'interface utilisateur finale (FUI). Effectuer une transformation d'**abstraction** sur les directives touchant à la fois l'accessibilité et l'utilisabilité (comme mis en avant dans l'article), permettrait de générer une nouvelle interface respectant ces contraintes.

D'un autre côté, certaines tâches données par les auteurs aux participants de l'étude ont menées à des problèmes communs d'accessibilité et d'utilisabilité. Ces tâches peuvent donc être un point d'entrée dans le niveau d'abstraction le plus élevée de CAMELEON, ***Task and Concepts level***. La volonté étant d'effectuer des transformations de **concrétisation** pour arriver sur une interface utilisateur finale où les tâches pourront être effectuées par les deux panels d'utilisateurs (déficients ou non).

Le *framework* CAMELEON s'attache à définir un ou des contextes d'usages qui suivent les quatre phases d'abstraction du système. Dans notre cas, le contexte d'usage évolue en fonction du type d'utilisateurs : déficients ou non-déficient. En effet, l'environnement sera différent : un utilisateur déficient visuel n'utilisera pas la solution web en marchant, et préfèrera un environnement calme, pour effectuer une navigation avec un lecteur d'écran. La plate-forme cependant se modulera peu entre les deux contextes d'usages, car les solutions de lecture de contenu web existent sur une multitude de supports.

Un rapprochement entre l'IDM et l'IHM pourrait donc permettre de faire ressortir des modèles prenant en compte les problèmes d'accessibilité présent actuellement dans le solutions web. Ces modèles pourraient permettre aux développeurs de toucher un public plus large sans nécessiter une adaptation de l'interface, et aux utilisateurs d'observer une uniformisation des solutions liées à l'accessibilité.

Étude technologique

Cross-Platform : PhoneGap / Steroids.js

Les technologies Cross-Platform Mobile fleurissent de nos jours. Ces solutions permettent de déployer un code sur plusieurs plate-formes mobiles (iOS, Android, Windows Phone, Blackberry, etc…). Nous nous attacherons à présenter deux solutions open-source : PhoneGap, fondateur et plus grand acteur du Cross-Platform, et la bibliothèque Steroids. Basé sur PhoneGap, ce framework tend à réduire les performances et à simplifier le déploiement de code.

PHONEGAP

PhoneGap est une solution développée à l'origine par Nitobi Software, puis par Adobe Systems. En 2011, Adobe donne le projet à Apache, qui le renomme Apache Cordova. On assiste alors à un changement d'orientation de la part de PhoneGap : l'équipe Cordova s'occupera désormais de développer la base du système, et l'équipe PhoneGap s'étendra sur l'implémentation d'outils complémentaires (par exemple, Adobe PhoneGap Build).

Les systèmes d'exploitation mobiles (principalement iOS, Android et Windows Phone) présentent chacun un langage de programmation qui est propre au constructeur, des *guidelines* et *code style* différents… Bref, le développeur doit apprendre en profondeur chaque SDK pour pouvoir créer des applications mobiles d'une certaine qualité, ce qui induit un coût de formation élevé. Surtout si l'objectif est de déployer une application multi-plateformes !

La volonté initiale de la plate-forme PhoneGap est de réduire l'écart entre les téléphones. Elle propose de développer pour les plate-formes mobiles en utilisant les langages du Web : HTML, CSS et JavaScript. En effet ces technologies, de part leurs maturité et leur améliorations constantes, permettent de se rapprocher des résultats obtenus grâce aux SDK mobiles propriétaires.

De plus, PhoneGap met à disposition un catalogue de plugins, qui sont des composants développés dans le code des plate-formes natives, qui permettent d'utiliser des fonctions spécifiques (capteurs du téléphone, notifications, in-app purchases, etc…).

Enfin, Adobe a mis à disposition *Adobe PhoneGap Build,* un service **payant** permettant de compiler, déployer, et partager ses applications dans leur cloud, pour gagner en rapidité et en utilisabilité.

Intéressons-nous plus particulièrement aux étapes de la mise en oeuvre d'une application HelloWorld sur PhoneGap :

Pré-requis : NodeJS doit être installé sur l'ordinateur, ainsi que les SDK cibles sur lesquels l'application sera déployée (iOS, Android…). *À noter que pour compiler une application iOS avec PhoneGap, il faut être en possession d'un Mac.*

1. Installation de la dépendance Node.js : PhoneGap (coût : 1 ligne de commande)
2. Création d'un environnement de travail (coût : 1)
3. Build du projet (coût : 1)
4. Lancement sur un émulateur (coût : 1)
5. Des fichiers XML de configuration doivent être écrit, un pour chaque cible. Ces fichiers contiennent le domaine de l'application, les auteurs, des méta-données sur la solution, et les **domaines extérieurs** auxquels l'application se doit d'accéder (capteurs + plugins développés en code natif - coût: ++).
6. Déploiement du projet (coût : 1)

Nous remarquons une mise en oeuvre extrêmement simple, facile d'utilisation et complète (il est tout aussi rapide de déployer les applications sur les stores). De plus, retrouver des langages de programmations très connus accélère la mise en route d'une app. PhoneGap est open-source et gratuit, un grand point positif aujourd'hui car le support en est décuplé, et sa gratuité tend un panel de développeurs à se rapprocher de cette plate-forme.

Le système de plugins proposé est quant à lui à double tranchant : avec un peu de chance les développeurs trouveront la totalité de leurs besoins dans le catalogue existant. Dans la majorité des cas cependant, ils devront créer des plugins par eux-mêmes, ce qui mène à écrire du code sur les plate-formes natives, et donc de devoir apprendre les syntaxes, les comportements, les fonctionnalités disponibles, etc…

Un développement 100% PhoneGap conduit indubitablement vers une IHM de site mobile, et non d'application native. Il faut utiliser des libraires externes pour faire croire à l'utilisateur qu'il navigue sur une « vraie » app (jQueryMobile, OnsenUI…). Ce problème vient de la quasi inexistence de composants d'interfaces ressemblant à la bibliothèque graphique fournie par les environnements natifs (design des boutons, barre de navigation, transitions et animations built-in…).

Enfin, des études de tests effectués démontrent que les technologies web portées sur mobiles ne sont pas « encore » aussi performantes que du code natif.

PhoneGap reste donc aujourd'hui encore incomplet vis-à-vis des fonctionnalités proposées par les plate-formes natives. On observe également un temps de latence pour la mise à jour des plugins, développés par des équipes extérieurs à PhoneGap/Cordova.

STEROIDS

Développé par AppGyver. Steroids est une plate-forme HTML5 qui promet des performances natives. Se basant sur Cordova/PhoneGap, cette librairie promet un débogage plus simple et une distribution plus rapide sur les différentes plate-formes.

Par exemple, avec Steroids, la nécessiter de s'inquiéter des profils d'approvisionnement. Les profils d'approvisionnement sont des fichiers permettant aux constructeurs (iOS, Android, Windows Phone…) de vérifier les auteurs de l'application et le niveau de leur compte (licence payante pour iOS par exemple), avant d'installer la solution sur le téléphone. Avec ce framework, il suffit à un client de scanner un QR code pour pouvoir installer l'application sur son téléphone.

Une autre fonctionnalité inexistante dans PhoneGap est l'ajout du concept de *hot code reloading*. Cela permet aux développeurs de ne pas avoir besoin de recompiler l'application pour observer les changements de code sur le/les téléphone(s) connectés.

Ces lots de fonctionnalités permettent par conséquent de gagner du temps et d'augmenter sa productivité.

Steroids propose également une plus grande ouverture au niveau des plugins. PhoneGap n'en supportent officiellement qu'un petit nombre, qui ont été sélectionné et vérifié, ce qui induit un catalogue avec peu de contenu. Steroids s'attache à simplifier les processus pour la validation et le déploiement de plugins.

Utilisant la plate-forme PhoneGap, la mise en oeuvre de Steroids est aussi simple et rapide. Niveau technologique, on observe l'ajout de la technologie Web Angular.js. Ce framework très intéressant va permettre de profiter de sa puissance au sein des apps, pour le coût d'un apprentissage de ses fonctionnements, parfois peu « naturels ».

Le plus gros avantage comparé à PhoneGap est la simplification du déploiement. En effet la fonctionnalité Adobe PhoneGap Build propose quelque chose de similaire, mais de façon payante.

L'utilisation de technologies de l'état de l'art induisent une grosse amélioration des performances par rapport à des solutions 100% PhoneGap. La sensation de naviguer dans un environnement natif est beaucoup plus évidente, notamment grâce aux composants graphiques fournis par Steroids. De plus, cet outil propose des animations et des transitions en tout point identiques à celles fournies par des applications développés avec les SDK propriétaires.

Cependant, Steroids est une plate-forme qui reste encore très jeune (lancée le 20 aout 2013), ce qui induit une liste de bugs clairement identifiés par ses créateurs. Par exemple, pour le moment, tout les plugins PhoneGap ne sont pas compatibles avec le fonctionnement interne de Steroids.

À la manière de Adobe PhoneGap Build, Steroids propose donc un déploiement plus aisé des solutions développés, dans un service cloud. Cela induit la nécessité d'avoir ce service pour ajouter des plugins, des icônes d'application, ou encore faire un package de la solution. La compilation/déploiement de projets Steroids dépendent donc **complètement** de leur service cloud, un risque que nombre d'entreprises refusera de prendre.

Comme PhoneGap, on observe avec Steroids des lenteurs pour la mise à jour vers les nouveaux systèmes natifs. Par exemple, à l'heure d'aujourd'hui, la géolocalisation et la caméra ne fonctionnent pas avec le nouveau système iOS8.

Enfin, Steroids privilégie une approche multi-pages, ce qui augmente la complexité du projet développé. Cet inconvénient est du à la volonté des créateurs de fournir des animations se basant sur celles proposés par les SDK iOS ou Android. Par exemple, afficher une fenêtre modale va effectivement apparaître de la même manière que dans une application native, mais va conduire à créer une nouvelle **WebView.** La multiplication de transitions et d'animations amèneront à une grosse charge pour des mobiles peu puissants, et demanderont un usage de bande passante réseau (3G ou Wifi) importante.

POSITIONNEMENT

Contexte d'usage

Utilisateurs : les deux solutions concernent des informaticiens souhaitant développer des solutions sur les plate-formes mobiles, en utilisant les technologies du Web : HTML, CSS et JavaScript.

Supports : Le supports visés sont mobiles (tablettes/smartphones), et (presque) indépendant des SDK fournis par les constructeurs.

Environnement : PhoneGap et Steroids sont des solutions qui veulent s'adapter aux mêmes environnements que lors d'un développement sur des plate-formes natives, ce qui induit par conséquent les mêmes problématiques : l'utilisateur mobile peut utiliser la solution dans une multitude d'environnements (en déplacement, footing, chez soi…). Cependant, les solutions cross-plateformes partagent une difficulté supplémentaire : la nécessité pour l'utilisateur de disposer d'une connexion internet satisfaisante, pour profiter des fonctionnalités de l'app (affichage dans une vue Web).

PhoneGap et Steroids proposent une tentative de **portage informatique** des SDK iOS et Android vers les technologies Web. La volonté des technologies cross-platform est également de rendre disponible les IHM fournies par les SDK natifs (composants de l'interface, interactions, transitions, animations).

D'un point de vue de la plasticité, les solutions cross-platform sont positionné au moment de la **conception**, pour faciliter la vie du développeur. En effet, PhoneGap permet de porter son code aussi bien sur Android que sur iPhone/iPad. Cela induit une réutilisation maximale des composants implémentés.

PhoneGap et Steroids présentent des solutions ad hoc pour répondre à des besoins en plasticité : réutiliser des composants d'interface sur différents supports, et permettre une migration **dirigée** par l'utilisateur.

Avis sur les solutions cross-platform

Avantages :

- **Simplicité** : le besoin d'apprendre en profondeur le SDK de chaque fournisseur disparaît. Une connaissance approfondie des langages Web est suffisante.

- **Rapidité** : la simplicité énoncée lors de la mise en oeuvre de PhoneGap ou Steroids induit une prise en main plus rapide, et l'aboutissement à une solution fonctionnelle dans des délais plus courts.

- **Déploiement aisé** : le déploiement sur les supports est souvent un point noir dans le développement mobile. Les solutions multi-plateformes en ont profité pour proposer des solutions innovantes et performante. On citera le service *Adobe PhoneGap Build*, ou Steroids, qui permet une distribution des solutions en quelques secondes.

Inconvénients :

- **Ne convient pas sur des gros projets** : en effet il a été souligné que ces solutions présentent un délai (compréhensible, certes) pour permettre l'utilisation de nouveaux composants proposés par les SDK natifs. Imaginons un projet d'application étendu sur 36 mois : l'attente vis-à-vis de ces solutions, qui restent open-source, fera perdre des jours de développement pour avoir une application « à jour ».

- **Gestion des performances** : Utilisant les technologies Web, les environnements de tests fournis n'ont évidemment pas d'accès à l'ensemble des données du support (utilisation du processeur, mémoire RAM utilisée, nombre de lectures / écritures, etc.). Il est par conséquent difficile de déterminer les performances / charges mémoire réelles de la solution développée.

- **Fluidité** : l'affiche de l'IHM dans une vue Web (WebView) induit un temps de réponse plus long que lors de l'utilisation de composants natifs. Même si les dernières avancées technologiques tendent à réduire ce délai, des différences de l'ordre des mili-secondes restent perceptibles pour l'utilisateur (un oeil humain est capable de différencier une animation durant 0.6 ms d'une autre durant 0.4 ms)

- **Dépendance vis-à-vis du réseau** de l'utilisateur : Si l'utilisateur ne dispose pas d'une connexion internet, les solutions cross-platform, par défaut, n'ont pas la possibilité d'afficher du contenu hors ligne. C'est un problème qui réduit beaucoup l'utilisabilité de ces librairies de développement.