BEYOND RESPONSIVE DESIGN: CONTEXT-DEPENDENT MULTIMODAL AUGMENTATION OF WEB APPLICATIONS

Plasticité des Interfaces – Flavien BOSSIAUX (SI5/IHM)

INTRODUCTION

Comme son nom l'indique, ce projet a pour but d'aller au-delà du responsive design, cantonné à adapter l'interface uniquement aux écrans en affichant, cachant et modifiant des éléments. L'idée ici est d'ajouter à tous types d'applications web, qu'elles soient prévues pour, ou non, une multi modalité qui interviendra selon le contexte.

En l'occurrence, ici on mixera la modalité graphique de saisie et d'affichage, avec la modalité vocale, pour la saisie en utilisant la reconnaissance de voix, et l'affichage avec une synthèse vocale.

Cette adaptation se fera grâce à différents capteurs comme les mouvements de l'utilisateur avec une ceinture adaptée, du Bluetooth, de la luminosité, du bruit ambiant, etc...

Quel contexte d'usage ?

Les utilisateurs

Pour ce qui est des utilisateurs développeurs, il n'y a aucune plasticité. L'outil ne s'adapte pas au développeur.

En revanche, pour ce qui est des utilisateurs d'une application web augmentée grâce à cet outil, on note énormément de plasticité, ce qui est le but même de l'outil. En effet, l'outil a été réalisé afin de pouvoir réagir à la fréquence cardiaque, de respiration, et mouvements de l'utilisateur, à condition qu'ils portent les capteurs adaptés sous forme de ceinture.

Un exemple où la vraie plasticité intervient, c'est si l'utilisateur est en mouvement, le téléphone vibre et proposera une adaptation de l'interface, que l'utilisateur aura le choix d'accepter ou non. Si oui, alors l'outil rajoutera une nouvelle modalité : la modalité vocale, et prendra soin de dicter le site à l'utilisateur, supposé occupé puisqu'en mouvement. Bien que l'interface graphique reste la même dans ce cas-là, on rajoutera de la multi modalité à cette interface, qui en devient donc plastique grâce à l'outil.

Les plateformes

Ici, l'idée a été de réaliser l'outil avec un langage de modèle, ici MARIA, qui fournit un langage abstrait, utile pour les modèles et essentiel pour l'adaptation, mais également un langage concret pour les différentes modalités (affichage graphique, voix, les deux, …). L'intérêt de rester à une abstraction élevée est justement de se séparer de l'implémentation finale, et donc de pouvoir être utile sur un maximum de plateformes différentes.

À la conception, une certaine plasticité est offerte dans le choix de l'implémentation, dans le papier on a choisi Android.

Pour ce qui est du changement de plateforme pas les utilisateurs, il est possible d'ajouter une règle proposant une adaptation de l'interface en fonction de celle-ci, bien qu'ils ne l'aient pas fait eux-mêmes. On peut tout de même parler de plasticité, car l'implémentation de cet outil se fera en amont, en dehors et indépendamment de la plateforme, bien qu'il paraisse évident que certaines adaptations comme la détection du bruit ne fonctionnera qu'en présence d'un micro.

L'environnement

C'est plutôt sur cette partie que l'implémentation de l'outil s'est tourné. En effet, il permet d'adapter les modalités en fonction de bruit ambiant, il ne proposera donc pas de saisies et synthèse vocale dans ces conditions, alors qu'il pourrait le faire dans un endroit silencieux.

Il a également été implémenté la détection du niveau de lumière, mais n'a pas été testé auprès des utilisateurs. On peut imaginer un changement de l'interface graphique, ou de la luminosité de l'écran, mais un contexte lumineux / non-lumineux a peu d'influence sur une autre modalité comme la saisie / synthèse vocale. Il en est de même pour la température.

Il utilise également le capteur Bluetooth, en regardant la puissance du signal. Cela sera utilisé dans les tests en définissant un certain émetteur Bluetooth comme étant celui de la voiture. Lorsque l'utilisateur rentre dans sa voiture, l'application cachera tous les éléments graphiques de saisies et les proposera vocalement afin que l'utilisateur garde les mains sur le volant.

Évidemment, cet outil permet une grande plasticité pour ce qui est de l'environnement, car c'est exactement à ce problème qu'ils souhaitent s'intéresser en ajoutant, enlevant ou mixant une nouvelle modalité, ici la voix, à l'affichage graphique.

QUEL MOMENT ?

À LA CONCEPTION

Pour ce qui est des développeurs, l'intérêt est très plastique, puisque leur but est de pouvoir intégrer facilement l'outil à tous types de projets, aussi bien ceux ayant prévus une multi-modalité, que d'autres. Il est question de rendre la chose la plus indépendante de l'application, le seul lien étant les technologies web, afin que le système puisse parcourir le DOM.

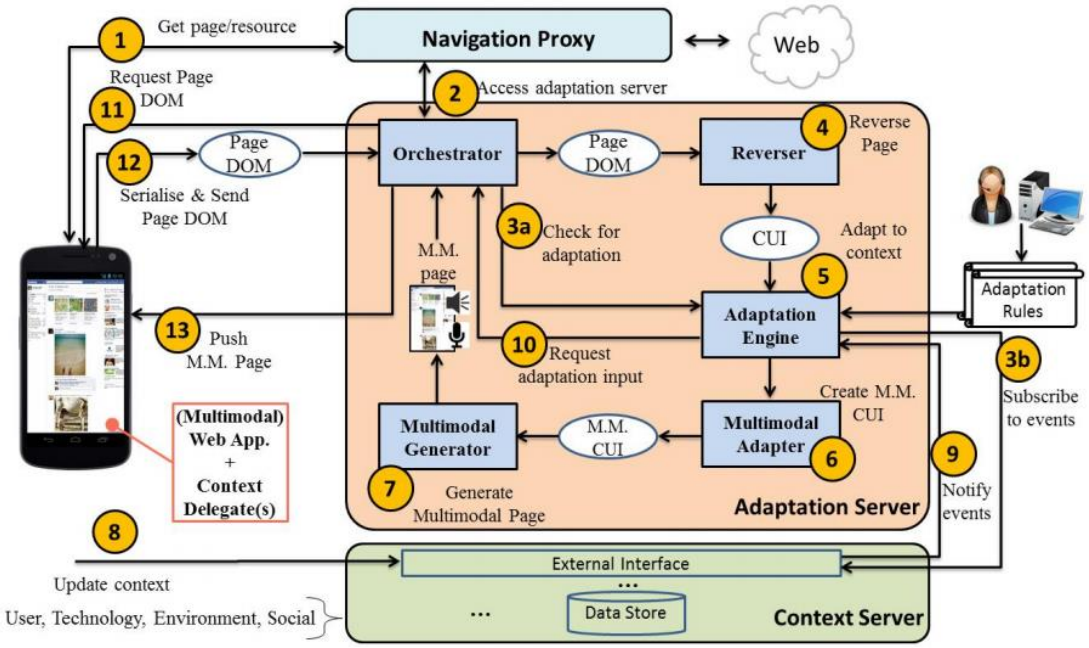
En effet, le déploiement est tellement séparé du reste de l'application qu'il pourrait prendre place en dehors de l'application, hébergé sur une autre machine, ce qui est d'ailleurs le cas sur la vidéo de présentation du projet. (http://youtu.be/7Y670aWNUDM).

À L'EXECUTION

C'est surtout à ce moment que la solution prend tout son sens. En effet, le principe est à l'ouverture de l'application par l'utilisateur d'écouter en permanence des données capteurs pour créer un contexte, et pour chaque changement de contexte, regarder s'il a été prévu une règle spécifiant de modifier les modalités en cours.

On parle donc de plasticité, car l'interface globale, et dons ses modalités sont plastiques. En revanche, les modalités mêmes, comme l'affichage graphique, ou la voix n'est pas plastique. C'est-à-dire que la solution ne se charge pas de modifier l'affichage, mais de l'enrichir ou appauvrir selon le contexte d'une nouvelle modalité, comme la voix.

QUELLE SOLUTION ?



1. On accède à l'application web via le Navigation Proxy, qui l'enrichit (modifie/rajoute les liens/références). L'utilisateur ne voit aucune différence.
2. Il fait suivre le DOM modifié à l'Orchestrator, qui est un module du serveur d'Adaptation
3. A) L'Orchestrator demande au module Adaptation Engine pour regarder si une adaptation est attendue (en fonction des règles d'adaptation définies et du contexte). Si une adaptation est requise, alors l'Orchestrator envoie le DOM au module Reverser, sinon il le renvoi au Navigation Proxy qui le renverra à l'utilisateur sans modifications.
   B) Lors du premier accès, l'Adaptation Engine écoute les évènements venants du contexte (sur le Context Server) en fonction des règles d'adaptation associées à l'utilisateur et l'application.
4. Le Reverser crée une description concrète du DOM (CUI) et l'envoie à l'Adaptation Engine.
5. L'Adaptation Engine identifie les adaptations à réaliser en prenant en compte les règles d'adaptations.
6. Si une adaptation est requise, on envoie le tout au Multimodal Adapter, qui se chargera de convertir notre description concrète CUI en description concrète multimodale (M.M CUI).
7. Le MultiModal Generator récupère la MMCUI et l'implémente, on retrouve donc une interface finale que l'on renvoie à l'Orchestrator, qui la fera suivre au Navigation Proxy, puis à l'utilisateur.

8. Si le contexte change, alors on envoie une requête au serveur de contexte (grâce à une API REST).
9. Si l'Adaptation Engine écoute les évènements du serveur de contexte, alors il reçoit l'évènement
10. Il contacte ensuite l'Orchestrator pour récupérer le DOM
11. L'Orchestrator demande le DOM au navigateur
12. Le navigateur sérialise le DOM (grâce aux scripts injectés étape 1), et le renvoie à l'Orchestrator, puis on répète les tâches 4 à 7

Avis personnel, avantages et inconvenients

Je suis personnellement assez mitigé sur cette solution. En effet, on peut y voir beaucoup d'avantages, et adapter une interface en fonction de son contexte est évidemment une bonne idée. L'idée est d'autant plus séduisante quand on voit que ces outils peuvent rendre accessible l'application à des non-voyants par exemple.

Le fait de séparer dans le principe même l'outil de l'application elle-même fait sens et laisse assez de liberté pour être décliné sur toutes les plateformes possibles.

L'étude partage également des résultats de tests utilisateurs dont la 2e a été assez concluante bien que certaines remarques assez pertinentes, comme le fait qu'on demande l'accord de l'utilisateur via le tactile de passer en mode vocal, car on en a déduit qu'il ne pouvait plus toucher l'écran. Il est également noté que la reconnaissance vocale est également source de certains problèmes à cause des différents types d'accents, mais ce n'est pas inhérent à la solution elle-même.

Au niveau de la plasticité, je n'ai absolument aucun argument qui irait à l'encontre d'un outil comme celui-ci.

Là où je reste sceptique, et qui remet un peu en cause l'essence du projet, et que ces applications sont le plus souvent dédiés pour des appareils mobiles. En effet, un ordinateur sera, à mon sens, moins susceptible de changer de contexte qu'un dispositif mobile. L'autre chose qui me fait aller dans ce sens est que pour pouvoir définir un contexte il faut des capteurs, qui sont le plus souvent présents sur les dispositifs mobiles. Mais c'est justement ici qu'il y a un problème, puisque si on parle de dispositifs mobiles, on peut se poser la question du cout sur l'autonomie de l'appareil en question. Scanner en permanence les dispositifs Bluetooth, tout en récupérant les données de la ceinture de l'utilisateur, en captant la luminosité, le bruit etc… Je me permets de faire le rapprochement entre changement de contexte et mobilité car c'est un peu comme est illustré la solution dans leur vidéo de présentation, en allant de dehors à dedans, en allant prendre sa voiture par exemple.