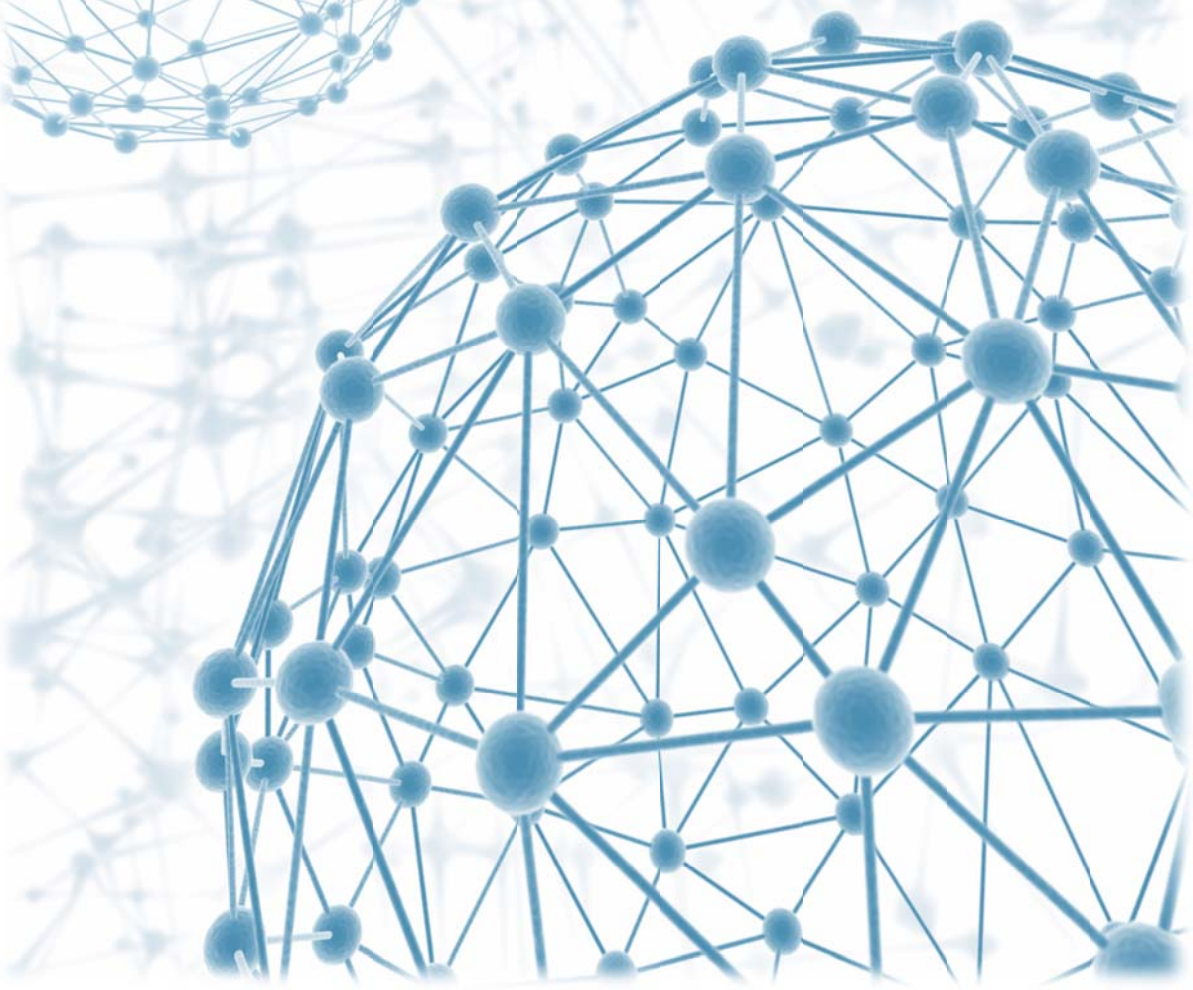
POLYTECH©

Adaptation des interfaces à l'environnement

Valentin POINSOT



Valentin Poinsot

Table des matières

I. TERESA

1. Contexte d'usage

Le contexte d'usage est le triplet <plateforme, environnement, utilisateur>.
Dans cet article, la plateforme regroupe les ordinateurs et les smartphones.
L'environnement utilisé peut se découper en plusieurs catégories :

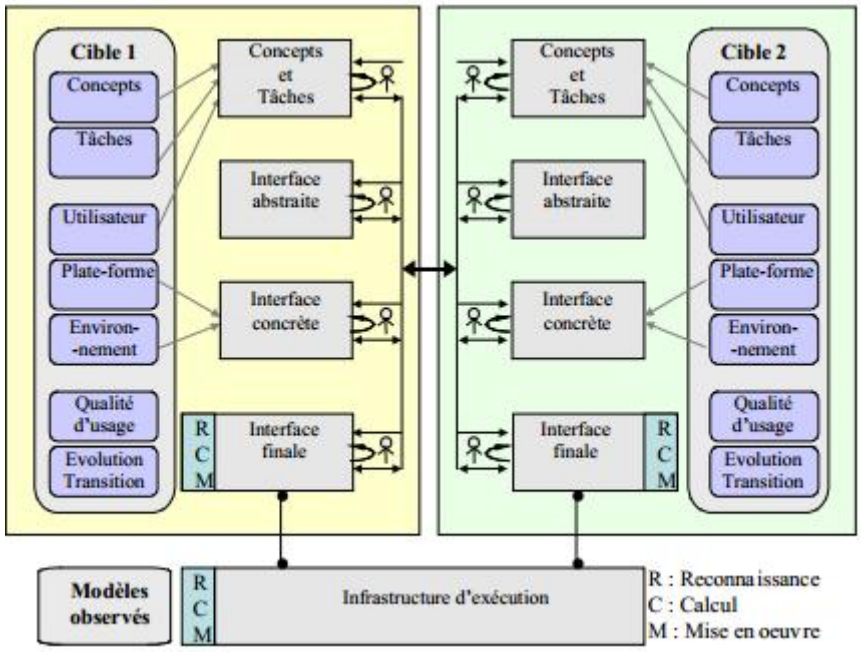
- Même tâche sur plusieurs plates-formes de la même manière.
- Tâches significatives que sur un seul type de plate-forme.
- Dépendances entre les tâches exécutées sur différentes plates-formes.
- Même tâche sur plusieurs plates-formes, mais réalisée de manière différente : différents objets du domaine, des objets de l'interface utilisateur, différents décomposition des tâches et avec des relations temporelles entre les sous-tâches

L'utilisateur visé par le projet TERESA est tout utilisateur utilisant un ordinateur et un smartphone et voulant pouvoir passer de l'un à l'autre pour la même application.

Le contexte d'usage changeant continuellement, il impose au système d'être capable de raisonner de sa propre conception à l'exécution : il ne s'agit plus seulement de percevoir le contexte d'usage et de commuter vers l'IHM préfabriquée la plus appropriée, mais d'élaborer une IHM conforme aux besoins de l'utilisateur et compatible avec le contexte d'usage courant. Dès lors, la fonction d'adaptation prend de l'ampleur. Elle pose des problèmes d'ordre algorithmique (la composition d'IHM par exemple) et de l'ordre de l'interaction. En effet, le choix du fait que l'adaptation doit se placer sous le contrôle de l'utilisateur ou non doit être fait.

2. Modèle sous-jacent

Le modèle sous-jacent que l'on peut ressortir de cet article peut se résumer grâce à l'image ci-dessous :



Les différentes étapes peuvent être expliquées comme ceci :

- Concepts et taches : Les tâches utilisateur et concepts du domaine décrivent l'interaction Homme-Machine selon une perspective du domaine, sans avoir besoin d'une quelconque représentation. Les différentes tâches qui doivent être effectuées afin d'atteindre les objectifs de l'utilisateur sont prises en compte avec les objets devant être manipulées pour leur performance. Souvent, ces tâches sont représentées hiérarchiquement avec des indications sur les relations temporelles entre elles et avec leurs attributs associés.
- Interface abstraite : L'interface abstraite structure l'IHM en espaces de dialogue et fixe la navigation entre ces espaces. Une interface utilisateur abstrait est définie en termes d'un certain nombre de présentations abstraites, chacun d'eux identifiant l'ensemble des éléments de l'interface utilisateur perceptibles en même temps. Chaque présentation abstraite est composé d'un certain nombre d'objets, qui sont des objets abstraits d'interaction identifiés en fonction de leur sémantique
- Interface concrète : Chaque objet de l'interface abstraite est remplacé par un objet d'interaction concrète qui dépend du type de plate-forme et a un certain nombre d'attributs qui définissent plus concrètement son apparence et le comportement.
  L'interface concrète fait donc le choix d'interacteurs pour :
  - Les espaces de dialogue alors réifiés en fenêtres ou canevas dans le cas d'IHM graphique

- Le contenu des espaces de dialogue : pour les espaces de production, les tâches élémentaires et concepts du domaine sont classiquement spécifiés par des boutons radio, cases à cocher, texte, etc.

  Le contrôle est typiquement assuré par les boutons « Valider » et « Annuler ». La navigation entre espaces est classiquement incarnée par des séparateurs (espace ou trait) ou des objets de navigation (boutons, liens hypertexte, etc.).

- Interface finale : L'interface concrète se traduit par une interface définie par un langage de logiciel spécifique

Le passage d'un niveau d'abstraction à un autre se fait à l'aide de transformations :

- La réification est la transformation d'un modèle en un autre modèle plus concret, c'est-à-dire plus proche du code
- l'abstraction qui est l'opération inverse de la réification. Elle permet donc de partir d'un modèle concret pour générer un modèle plus abstrait
- la translation, c'est-à-dire l'adaptation d'un modèle vers un autre modèle du même niveau et qui permettent par exemple une adaptation partielle

3. Solution

Pour mieux comprendre ces niveaux, on peut considérer un exemple de tâche : faire une réservation d'une table d'un restaurant. Nous verrons donc les différentes étapes du modèle sous-jacent dans l'ordre.

- Concepts et taches

Cette tâche peut être décomposée en sélectionnant la date, l'heure d'arrivée dans le restaurant, le nombre de personne et, le cas échéant, en sélectionnant les préférences de la table. Puis le fait de hiérarchiser ces actions permet clarifier ces différentes actions, de vérifier que toutes les actions de l'application sont présente et de construire une interface abstraite plus facilement.

- Interface abstraite

Au niveau de l'interface utilisateur abstraite, il faut identifier les objets nécessaires pour soutenir l'interaction de ces tâches: par exemple, en précisant la date et l'heure d'arrivée dans le restaurant appelé des objets de sélection interactifs. Ainsi, dans cet exemple, la sélection de certaines actions (le choix du nombre de personne par exemple) peut être prise en charge par une liste d'objets. Ce choix est plus efficace qu'une case à cocher parce que le choix dans une liste prend en charge une seule sélection à partir d'une longue liste d'éléments potentiels alors que la case à cocher est adaptée pour supporter des choix multiples d'un nombre limité de possibilités.

- Interface concrète

Lorsqu'il faut passer à l'interface concrète, il faut tenir compte des éléments d'interaction spécifiques prises en charge par la plateforme. L'interface utilisateur de l'exemple est le résultat de ces choix et ceux impliquant des attributs spécifique à l'interface tels que le type de police, taille, le premier plan, couleurs de fond, images de décoration, etc…

- Interface finale

Une fois toute fois toutes ces étapes effectuées, il faut implémenter l'application dans le langage spécifique défini au début du projet.

4. Avis

Je pense que rendre chaque application applicable à plusieurs plateformes est une bonne chose. En effet, la diversité des plateformes existantes est de plus en plus importante et ce sont les applications à s'adapter à celle-ci et non aux utilisateurs à les posséder toutes afin de bénéficier des applications l'intéressant. De plus le fait de posséder les transformations permettant de passer d'une étape à l'autre du modèle est très intéressant car pour un travail en groupe, où chaque personne possède sa manière de travailler sur une IHM, cela permet de mettre en commun les travaux de chacun sans difficulté.

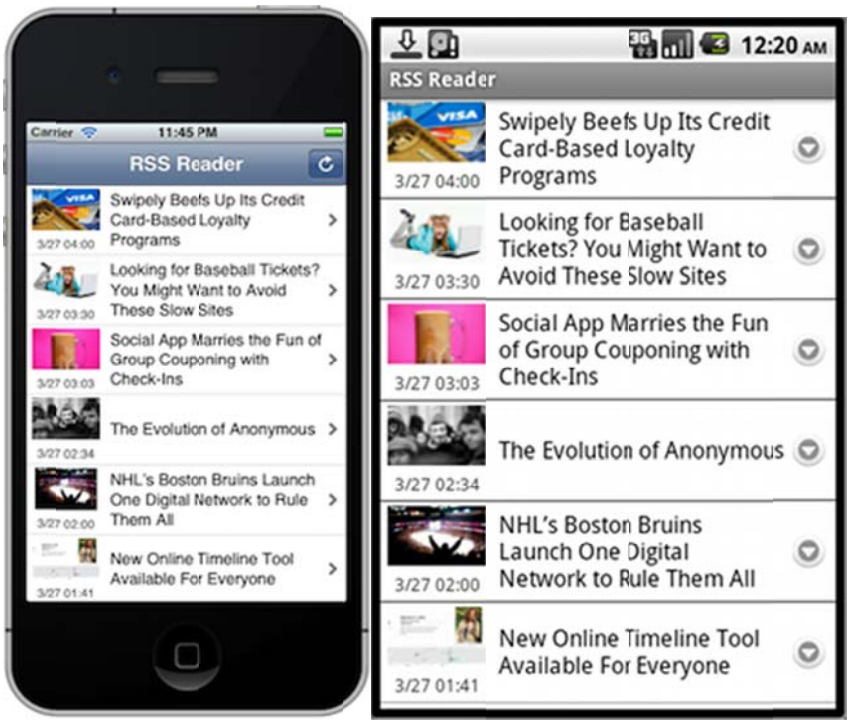
II. Comparaison de technologies

1. Appcelerator (Titanium)

Appcelerator est un environnement ouvert, extensible pour créer des applications natives à travers différents appareils mobiles et systèmes d'exploitation, y compris iOS, Android, BlackBerry ainsi que des hybrides et HTML5.

Appcelerator fournit une machine virtuelle permettant de développer des applications natives en JavaScript.

Un exemple utilisant Appcelerator :

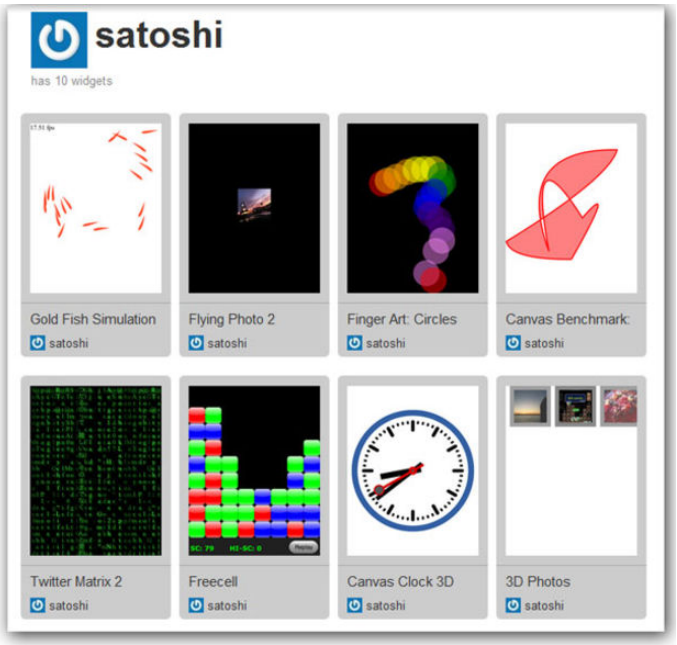


2. WidgetPad

Il s'agit d'un IDE en ligne, gratuit et open-source, avec lequel les développeurs peuvent créer et distribuer sur les « stores » facilement, des applications pour smartphones, tout en respectant les standards web et ses technologies (HTML 5, CSS, et Javascript). WidgetPad a pour objectifs de simplifier la création d'applications, en éliminant notamment le recours à des plateformes spécialisées.

Le créateur de WidgetPad, Satoshi Nakajima, croit que, avec le support d'entreprises telles que Google et Apple, le HTML5 va, à moyen ou long terme, supplanter les langages C++, Objective C, et Java.

Quelques exemples des possibilités de WidgetPad :



3. Comparaison

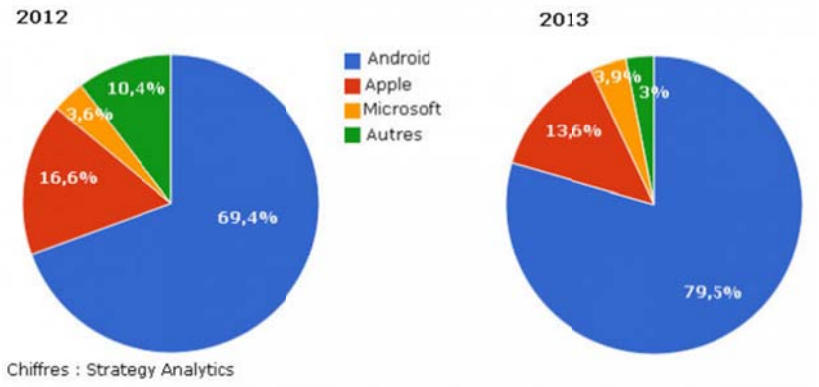
a) Points commun

Ces deux technologies utilisent toutes deux les technologies du web, à savoir HTML 5, CSS et JavaScipt. Cela permet aux développeurs de ne pas à avoir à apprendre les langages spécifique aux différentes plateformes mobiles. De plus elles permettent de créer des applications multi-média entièrement interactives. Le fait de pouvoir mettre leur application sur différent store (App Store, Google Play, …) permet de toucher un plus grand public.

Enfin, elles permettent l'utilisation d'un code unique pour différentes plateformes ce qui offre un gain de temps et réduction des coûts de développement important.

b) Différences

La principale différence à noter est les différentes plateformes supportées par ces technologies : Appcelerator Titanium ne supporte actuellement qu'iOS, Android et Tizen, mais le support de BlackBerry 10 est en développement et celui de Windows 8 est envisagé. WidgetPad n'a supporté qu'iOS dans un premier temps car WidgetPad s'est développer en 2009 et iOS dominait le marché du mobile. WidgetPad s'est ensuite étendu à Android. En revanche en constatant qu'iOS et, surtout, Android possèdent les deux plus grandes parts du marché mobile en 2013, les deux technologies ciblent plus de 90% des utilisateurs.



Au niveau des ressources utilisées, Appcelerator utilise entièrement les capacité des différents OS qu'il supporte.

Si WidgetPad exploite pleinement les capacités d'iOS (GPS, accéléromètre, compas,…), il est nettement moins performant sur android. En revanche Appcelerator tire sa force sur ce point : il utilise pleinement les capacités des plateformes qu'il supporte.

4. Avis

Je trouve qu'il existe d'innombrables avantages au fait de développer une application qui peut fonctionner sur toutes les plateformes majeures. En revanche si notre application exige une grande utilisation d'API d'un système d'exploitation particulier, les logiciels de cross-platform risquent de freiner la conception. En plus de cela, Android et iOS publient de nouvelles versions de leur OS presque chaque année ce qui peut également poser problème.

L'autre point qui peut freiner l'utilisation de ces technologies est qu'elles sont toujours en version beta pour la plupart d'entre elles voir dans des versions encore plus précoce.

L'avantage est que ces technologies utilisent plus ou moins le même langage à savoir HTML 5, CSS et JavaScript ce qui permet de passer d'un logiciel à l'autre sans problème.