UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE
INSTITUT CUEEP
DESS Multimédia et Internet pour le Commerce Electronique

CORBEAUX Pauline
RESCHKE Romain

Tuteur : José Rouillard

Projet de DESS

Pour la plasticité des interfaces

Année 2003 / 2004

Sommaire

Remerciements

Nous tenons à remercier Mr Yvan Peter pour les cours qu'il nous enseignés en XML et XSL. Ceux-ci nous ont beaucoup servis durant toute la durée du projet.

Nous remercions également Mr José Rouillard, responsable du projet Plastic ML, pour son encadrement attentif. Il a toujours été disponible lorsque nous avions besoin de son aide. De plus, il a su nous documenter et nous guider lorsque nous nous égarions lors de l'étude des languages traitant la plasticité. Enfin, il nous a mis sur certaines pistes permettant d'avancer dans ce projet.

Introduction

Le Projet du DESS MICE constitue un travail de recherche et de développement. Il se déroule sur toute l'année universitaire et nous permet de nous familiariser avec de nouvelles technologies et de travailler par équipe de deux.

Au début de l'année, les professeurs nous ont présentés la liste des projets. Le sujet 5 nous a tout de suite intéressé. Il nous demandait de créer un langage de description abstraite d'interaction Homme-Machine basé sur XML. Ce langage, Plastic ML, permet de décrire des éléments d'entrée et de sorties d'interface, à un haut niveau, c'est-à-dire sans faire référence au périphérique qui sera utilisé lors de l'interaction. Grâce à des transformations XSLT, les documents en Plastic ML sont traduits automatiquement vers des langages idoines (XHTML pour PC, WML pour les téléphones WAP, VoiceXML pour les téléphones fixes et mobiles). Un modeleur graphique permet de créer des documents en Plastic ML, grâce à des manipulations directes (clavier/souris), sans avoir à écrire le code correspondant.

Le langage Plastic ML 1.0 existe déjà, notre projet consistait donc à l'améliorer et à créer une version 2.0. Pour cela, nous devions faire un travail de recherche d'informations qui consistait à étudier les différents langages existants gravitant autour de la plasticité des interfaces et à synthétiser les éléments communs pour peut-être intégrer leurs fonctionnalités dans la future version de Plastic ML. Puis, il fallait étudier la grammaire (DTD) de Plastic ML 1.0 pour y ajouter de nouveaux services.

La seconde étape du projet était de modifier le modeleur graphique (Toolkit) développé en Java pour qu'il intègre les nouvelles fonctionnalités de Plastic ML 2.0.
Enfin, il nous fallait travailler sur la personnalisation de l'interface en fonction du rôle de l'utilisateur.

1. Définition de la plasticité

De nos jours, le nombre de périphériques ne cesse d'augmenter. Or ces périphériques sont de plus en plus différents. Certains ont des écrans très petits, d'autres plus grands, les résolutions sont aussi différentes et tous permettent de lire des contenus provenant d'Internet, certains périphériques n'ont même pas d'écran. Actuellement, lorsque l'on doit faire une application Internet composée de N pages et que l'on a M périphériques, on doit créer N x M pages. On voit bien par cela que plus il y aura de périphériques, plus cela prendra de temps de créer une application multi-plateforme.
La plasticité des interfaces décrit une interface de manière abstraite. Si toutes les interfaces étaient créées de manière abstraite, on pourrait les transformer directement pour le périphérique demandé, on gagnerait alors beaucoup plus de temps dans le développement de l'application.

2. Brève description de Plastic ML

Plastic ML est un langage de description abstraite d'interaction Homme-Machine basé sur XML. Ce langage ML permet de décrire des éléments d'entrées et de sorties d'interfaces, à un haut niveau, c'est-à-dire sans faire référence au périphérique qui sera utilisé lors de l'interaction. Grâce à des transformations avec XSLT, les documents en Plastic ML sont traduits automatiquement vers des langages idoines (HTML pour le Web, WML pour le WAP, VoiceXML pour le téléphone) (cf. Figure1).

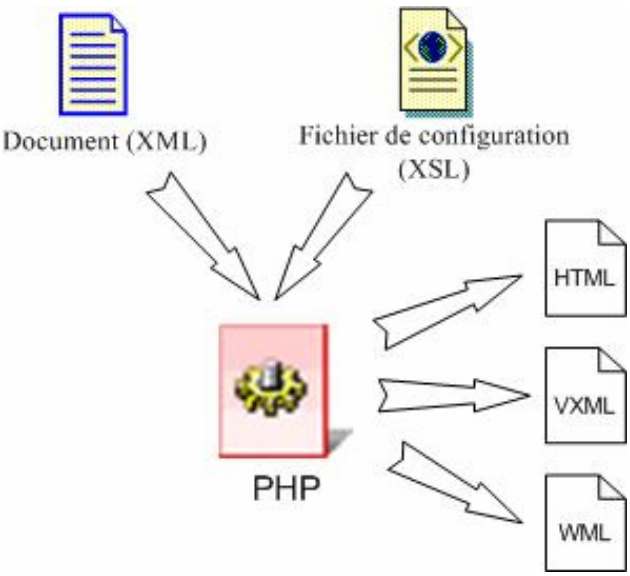


Figure1 : Traitement des données

La personne qui veut créer un fichier en Plastic ML 2.0 doit se baser sur la DTD de Plastic ML 2.0. Pour cela il peut utiliser par exemple XML Spy, éditeur de documents XML. Pour la version 1.0, Plastic ML est accompagné de son outil de développement Plastic ML Toolkit. Cette boîte à outils permet de créer, modifier, sauvegarder du code Plastic ML par manipulation graphique, et fournit aux concepteurs une transformation dans un langage à balises cible pour une utilisation concrète (HTML, WML, VoiceXML).

Un utilisateur sur son ordinateur voulant visualiser une page Plastic ML va se connecter au serveur HTML pour récupérer la page et certaines données dans une base de données. Un utilisateur ayant un téléphone portable pouvant aller sur le WAP pourra soit téléphoner et se connecter au serveur vocal VXML, soit aller sur le WAP et se connecter au serveur WML. Il aura donc le choix entre deux modes de connexion pour visualiser la page désirée (cf Figure2).

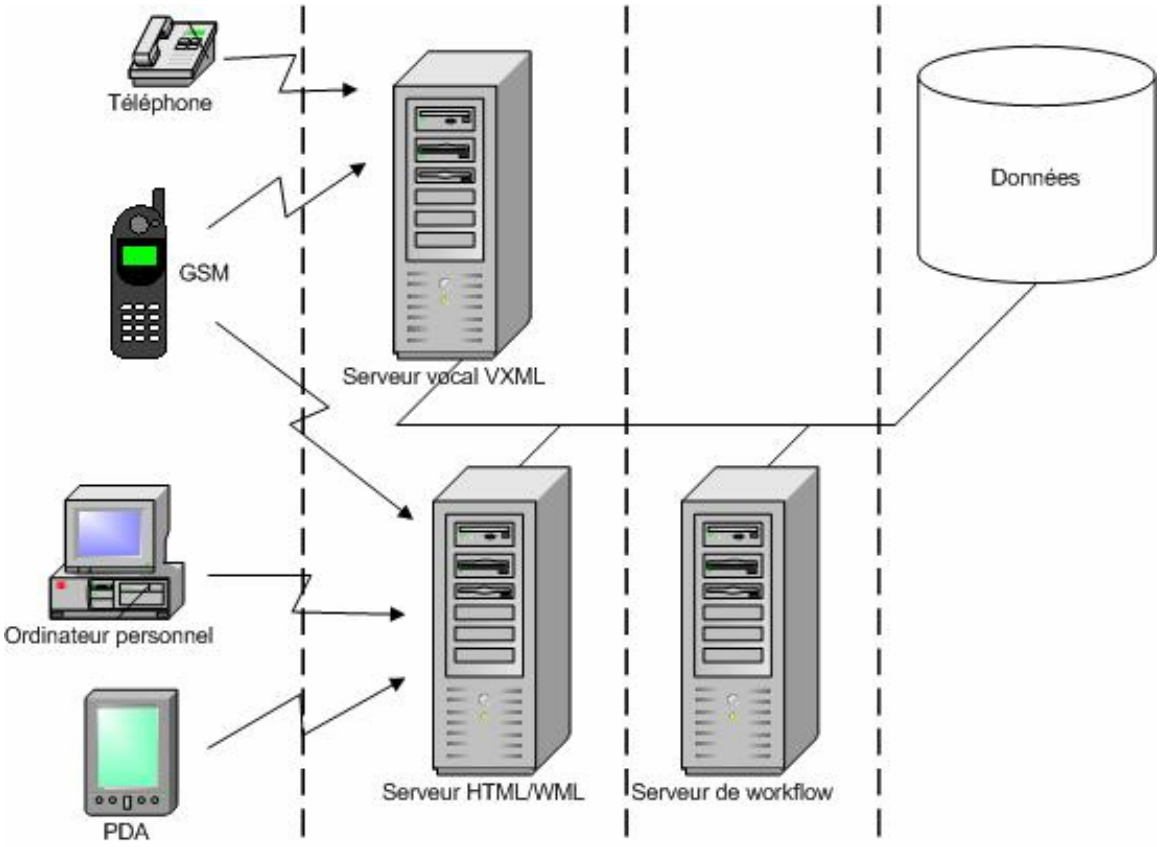


Figure2 : Architecture du système

Dans la version 2.0, la notion de rôle a été abordée. L'utilisateur acquiert un certain rôle en se connectant à une base de données. Par la suite, on affiche seulement à l'écran ce que cet utilisateur est autorisé à voir par rapport à son rôle. Cette notion de rôle est gérée dans une première transformation XSLT (cf Figure3). Grâce à la deuxième transformation du fichier Plastic ML, on affiche la page dans le langage désiré par l'utilisateur.

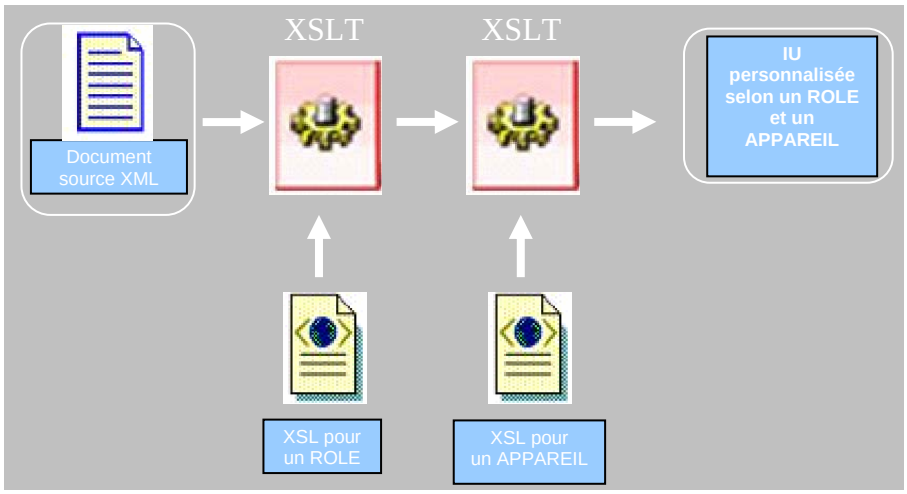


Figure3 : Le rôle d'un utilisateur

3. Les concurrents et leur évolution

3.1. Liste des concurrents

3.1.1. AUIML (Abstract User Interface Markup Language)

- ✓ **Définition :**

  AUIML est un langage de définition d'interfaces principalement développé par IBM. Ce langage est basé sur XML, donc il peut être développé sur n'importe quelle plateforme. Il permet le codage d'information indépendamment de l'appareil.

- ✓ **Caractéristiques :**

  AUIML permet de décrire l'interface en termes :
  - D'éléments manipulés (données initiales, structures et données complexes telles que les images ou du son).
  - D'éléments d'interaction : types simples mêlant de la structure et des fonctionnalités (choix, groupe, table, arbre) ;
  - D'actions : permet de décrire un micro-dialogue pour la gestion d'évènements entre l'interface et les données.

- ✓ **Positionnement par rapport aux autres langages de plasticité des interfaces :**

  - Le concepteur doit spécifier l'interface à la main.
  - On ne trouve aucune spécification pour ce langage et on peut voir que, lancé en 1998, ce projet n'a jamais évolué. On peut donc penser que ce projet a été arrêté.
  - Il devait ensuite pouvoir être transformé dans le langage approprié à l'appareil mais après avoir passé un certain nombre de phases, ce projet a abouti à un vocabulaire XML avant de se terminer.

3.1.2. Cameleon (Context Aware Modelling for Enabling and Leveraging Effective interactiON)

- ✓ **Définition :**

  "The CAMELEON Project (Context Aware Modelling for Enabling and Leveraging Effective interaction) is a Shared-Costs RTD three-year IST Project started in October, 1st 2001. The goal of the CAMELEON Project is to build methods and environments supporting the design and development of highly usable context-sensitive interactive software systems.
  In the CAMELEON Project several tools have been developed, two of them are publicly available: TERESA, VAQUITA. "

- ✓ **Caractéristiques :**

  "The main goals of CAMELEON Project is to support design and development of highly-usable context-sensitive interactive software systems mainly by:

- producing a development framework that incorporates and structures the development process using our models, techniques, architectures and tools;
- providing the means to express context-dependent information in a set of models usable at design time by developers and at run-time by dynamically reconfigurable systems;
- identifying criteria, methods and techniques for using information in abstract representation to drive the design and development of the concrete interface of heterogeneous devices while preserving usability;
- developping tools and components that allow designers to obtain systems represented in the above models.

These results will be tested in a number of case studies that institutions from different application domains have already agreed to provide to the CAMELEON consortium. The case studies will check the effectiveness and generality of the results of the project. The case study will consider a large range of interaction technologies:

- Graphical user interfaces for multiple platforms (such as the case study for home temperature control proposed by EDF);
- 3D applications (such as the BtoB case study proposed by IS3);
- Multimedia applications (such as the museum application proposed by I.S.T.I.)
- Applications where users can seamlessly interact with a dynamic number of devices (such as that proposed by Université Joseph Fourier).

As a result of the CAMELEON's activity the European industrial and academic community will be made aware of the potentials offered by model-based approaches to obtain usable multi-platform applications; tools supporting these approaches will be available and a number of pilot applications will be developed using such tools to exemplify these new potentials. ”

✓ **Positionnement par rapport aux autres langages de plasticité des interfaces :**

Ce projet n’est pas open source, il faut pouvoir entrer dans le consortium pour pouvoir accéder aux sources. Nous n’avons donc pas pu connaître les techniques employées et comprendre comment cela fonctionnait.

[Cameleon Project, plasticity of user interfaces, http://giove.cnuce.cnr.it/cameleon.html]

3.1.3. SunML (Simplified Unified Natural Markup Language)

- ✓ **Définition :**
  SunML est un langage développé par l'université Sophia Antipolis de Nice. Il permet à l'aide d'un seul fichier XML, de définir des interfaces utilisateurs. Un compilateur, *SunMLCompiler*, génère les interfaces directement utilisables par les appareils du type PDA, téléphone fixe ou PC.

- ✓ **Caractéristiques :**
  Actuellement, le compilateur existe et permet la génération des fichiers VoiceXML (pour le téléphone fixe), HTML (WAP et PC), Java Swing et Java AWT. La description des composants les plus basiques a aussi été faite.
  Il reste à décrire plus de composants et à gérer les événements.
  - Il comporte 19 balises mais est facilement évoluable : l'ajout de balise n'est pas compliqué (modification de la DTD, ajout du composant (fichier java), modification du parser).
  - La balise <style> intégrée à la balise mère <sunml> permet de personnaliser les composants utilisés dans la partie structure.
  - Les composants gérés par SunML sont : la fenêtre, le panel, le texte, le bouton (balise *action*), la liste.
  - La partie *extern* qui permet d'importer une classe Java.
  - La partie *behavior* décrit les comportements à appliquer aux composants (action sur le clic d'un bouton par exemple). On peut lui demander d'exécuter une fonction d'une certaine classe qu'on aura ajoutée dans la partie *extern*.

- ✓ **Positionnement par rapport aux autres langages de plasticité des interfaces :**
  Conceptuellement, SunML est supérieur à UIML car il nécessite un seul fichier XML à la source, alors que UIML nécessite un fichier par média.

3.1.4. UIML (Unified Interface Markup Language)

- ✓ **Définition :**

  UIML est un langage pour créer des IHMs pour n'importe quel périphérique, n'importe quel langage et n'importe quel OS.

- ✓ **Caractéristiques :**
  - Il utilise un petit nombre de balises à sémantique élevée (<part>, <property>,…).
  - Il inclut un moteur de rendu qui permet de générer à partir du fichier UIML un fichier WML, HTML et VoiceXML.
  - Il permet de définir la structure, le style et le comportement (behaviour) d'un document.
    - la structure décrit les composants à ajouter à la page
    - le style décrit graphiquement les composants (taille, titre, font pour les textes,…)
    - le comportement donne l'action à effectuer lors de l'occurrence d'un événement, pour un composant donné. Il s'agit en fait d'un appel à une sorte de méthode (non implémentée).

- ✓ **Positionnement par rapport aux autres langages de plasticité des interfaces :**
  - Malgré sa reconnaissance, UIML n'est pas encore un standard
  - Il faut un fichier UIML par plateforme ce qui nuit complètement à la notion de généricité du langage. Il est trop proche du Java.
  - UIML provoque de nombreux bugs en générant les fichiers VoiceXML.

3.1.5. XForms (XML Froms)

✓ **Définition :**

XForms, ou XML Forms, est une spécification de définition de formulaires, basée sur XML, destinée aux futures générations de HTML et XHTML.

✓ **Caractéristiques :**

- XForms est une spécification du W3C.
- Il sépare la présentation du contenu par une organisation en 3 sections bien séparées:
  - Les données du formulaire
  - L'interface utilisateur (aspect visuel)
  - La logique d'interaction
- Il permet d'utiliser des technologies comme XSL et CSS.
- XForms comprend deux parties, dont la première, le *XForms Data Model*, porte sur la modélisation des données, et dont la deuxième, non encore définie, portera sur l'interface utilisateur. Cette séparation facilite la réutilisation et permet de générer plusieurs présentations à partir d'un seul modèle de données.
- Il utilise XML à la fois pour définir les formulaires mais aussi pour transporter les données (avec Unicode) et peut aussi se coupler avec d'autres technologies comme WML et SVG (langage de graphisme vectoriel).
- Pour les formulaires XForms est supérieur à HTML grâce :
  - Au fort typage (les données entrées doivent avoir un certain type (par exemple pour saisir un prix, l'utilisateur ne pourra pas rentrer des lettres dans son champ de formulaire).
  - Au format XML : les données envoyées sont dans un fichier XML, donc le serveur ne doit pas réassembler les données du formulaire.
  - A l'internationalisation des données transmises.
- Comme les éléments de l'interface sont définis de manière abstraite, c'est la plateforme cible qui va choisir la manière dont elle va présenter le formulaire (par exemple, la balise <select1> de XForms permet de générer une liste déroulante en HTML et une suite de boutons radios en WML. Au final, la personne pourra choisir une seule valeur parmi plusieurs.

✓ **Positionnement par rapport aux autres langages de plasticité des interfaces :**

- Même si XForms est devenu un standard, il a du mal à s'imposer face à HTML qui est moins contraignant à programmer. Aucun éditeur de site Web ne permet à ce jour d'utiliser XForms.

3.1.6. XIML (eXtensible Interface Markup Language)

✓ **Définition :**

XIML est un langage basé sur XML de représentation d'interfaces multi plates-formes. Il couvre le cycle de vie entier d'une interface utilisateur: conception, développement, opération, gestion, organisation, et évaluation.

✓ **Caractéristiques :**

Ce langage se veut ouvert et extensible :
• Il est basé sur XML et n'impose aucun outil de spécification.
• On le considère comme un métalangage. Ainsi il sera aisé de le connecter à des outils ou de le mettre à jour pour générer de nouvelles IHMs.

Une description XIML est semblable à un dépôt centralisé de données. Elle est composée de trois piliers : le composant, l'attribut, la relation.

- Un composant est un élément décrivant les tâches, les concepts du domaine, les utilisateurs (aspects abstraits d'une interface), les présentations ou le dialogue (aspects concrets d'une interface).
- Un attribut appartient à un composant. A partir d'un ensemble d'attributs, il est possible de décrire les caractéristiques ou les propriétés d'un composant.
- Une relation décrit un lien entre deux ou plusieurs composants.

On utilisera l'outil MOBI-D pour passer d'une présentation abstraite à une présentation concrète.

✓ **Positionnement par rapport aux autres langages de plasticité des interfaces :**

La définition de ce langage est en cours de finalisation. Aucun outil n'est proposé pour gérer les différents niveaux de conception et la génération des IHMs.

[thèse de David Thévenin,

Adaptation en Interaction Homme-Machine: le cas de la Plasticité, 21 décembre 2001, http://iihm.imag.fr/thevenin/these/TheseDavidThevenin.pdf]

3.1.7. XUI (eXtensible User Interface)

✓ **Définition :**

XUI est un framework open source pour la construction d'applications légères mobiles et de bureau. Grâce à ce framework on peut économiser jusqu'à 60% du code typiquement nécessaire pour construire une application. Ceci a comme conséquence des vraies économies en temps d'élaboration et coûts d'entretien et va aussi considérablement améliorer la fiabilité et la stabilité de vos applications Java.

XUI est également un outil de développement idéal pour construire occasionnellement des applications connectées telles que celles qui sont nécessaires pour des field workers ou PDAs. Le framework XUI est conçu afin de permettre à des données d'être tirées et poussées vers diverses sources de données. Dans le package de base le réalisateur peut charger des données d'une simple source de données et sauvegarder les résultats dans une autre.

✓ **Caractéristiques :**

- Il est extensible, open source et donc gratuit.
- Basé sur Java
- Marche avec la JDK 1.1.4
- Supporte les machines virtuelles PDA
- AWT ou Swing
- Accès complet à API
- Description en page XML
- Seulement des pages XML ou seulement des pages Java ou un mélange de XML et Java
- Séparation de la logique, des données et du contrôle
- Feuilles de style XML (Stylesheets) pour contrôler l'apparence
- Manipulation simple d'événements
- Permutation (swapping) de pages
- Gestion de ressources
- Lien bi-directionnel de données

✓ **Positionnement par rapport aux autres langages de plasticité des interfaces :**

Pour l'instant XUI a un avantage que les autres langages n'ont pas ou peu : les feuilles de style (Stylesheets) qui permettent de contrôler l'apparence d'une

page et donc de personnaliser la page selon l'utilisateur. Ce projet est encore en cours. Cependant très peu de documentation nous est proposée sur Internet.

[Site de XUI, http://xui.sourceforge.net/]

3.1.8. XWeb

✓ **Définition :**
XWeb est un ensemble d'outils pour générer des sites Web complets grâce à un ensemble de fichiers d'entrée dans différents formats, plus précisément XHTML et vos propres formats XML. Il essaye d'atteindre les buts suivants:

- Séparation complète du contenu et de la répartition (layout)
- Représentation sémantique du contenu dans n'importe quel format XML
- Edition simple du site Web, y compris l'addition/suppression des pages

✓ **Caractéristiques :**

- Ecrit en Java.
- Traitement XSLT pour des fichiers d'entrée XML (y compris les chaînes de feuilles de style)
- Rendu des boutons et des bannières en utilisant soit le renderer interne, soit le SVG
- Ajout d'information approprié pour la structure de navigation dans le processus XSLT
- Open source.

Actuellement on peut écrire son propre code XSLT pour obtenir exactement les résultats que l'on veut ou l'on peut employer la feuille de style générique et on n'aura même pas à jeter un coup d'oeil à aucun code XSLT. Ce dernier est bien sûr moins flexible mais les feuilles de style offrent un certain nombre de paramètres pour contrôler les aspects de disposition générale comme la position et la structure de navigation et CSS peut être employé pour changer plus que juste les détails de l'aspect.

✓ **Positionnement par rapport aux autres langages de plasticité des interfaces :**

Ce langage est open source, il est ainsi assez facile de le récupérer.
Les processus principaux sont traités par un ensemble de classes Java qui ont des interfaces pour l'écriture. Ces outils sont conçus pour être flexibles et pour permettre l'écriture mais ils exigent quelques maîtrises dans les opérations de commande de ligne et l'édition XML/XSLT. Il n'y a aucun support spécifique à l'application.
Les feuilles de style génériques permettent aux utilisateurs d'éviter d'écrire du code XSLT eux-mêmes.
L'inconvénient est qu'il ne génère que du XHTML. XWeb est conçu pour être flexible et est ainsi complexe. Bien que XWeb ait tenté de rester simple il n'est pas

conçu pour des personnes sans au moins quelques connaissances de HTML et de CSS.

[site de XWeb, http://xweb.sourceforge.net/]

3.1.9. UIX (User Interface and eXecutive)

✓ **Définition :**

UIX est un ensemble de technologies qui constitue un framework pour construire des applications Web, c'est un dérivé du framework J2EE pour construire des applications Web. Il est basé sur le framework MVC (Model View Controller). UIX supporte les navigateurs Web et les appareils mobiles. UIX est basé sur la technologie Java, mais Java n'est pas obligatoire chez le client.
UIX inclut des bibliothèques de classe Java, APIs, des langages XML, et d'autres technologies pour développer différents aspects des applications basées web.

✓ **Caractéristiques :**

- Standard. UIX est basé sur les dernières versions des standards J2EE et XML. On peut intégrer facilement UIX avec des applications APACHE.
- Plateforme indépendante. Comme UIX est implémenté avec Java et d'autres technologies Web portables, UIX peut être utilisé pour développer des applications pour n'importe quelle plateforme et navigateur.
- UIX fournit un framework open source, flexible pour le développement. Possibilité de choisir toutes les fonctionnalités d'UIX ou seulement celles intéressantes pour l'application. Par exemple, on peut employer des composants d'UIX pour générer des pages, ou on peut employer ses propres pages HTML ou pages de serveur Java (JSP) pour générer tout en tirant profit toujours des dispositifs restants d'UIX. Accessoirement, on peut employer n'importe quelle technologie de données principale qui convient le mieux à ses besoins.
- IDE. La documentation d'UIX est intégrée dans l'IDE Jdeveloper. Pour faciliter la création de pages UIX, un éditeur de Schémas XML et une palette d'éléments font partie de l'IDE.
- Personnalisation. Les applications UIX sont facilement personnalisables pour différents environnements chez les utilisateurs finaux. On peut aussi appliquer différents layout et styles. On peut changer beaucoup d'aspects de l'application indépendamment, y compris la disposition de la page, les modèles, et la création d'images. L'environnement rend des personnalisations simples faciles, et des personnalisations plus compliquées possibles.
- Accessibilité et internationalisation. Les composants de l'interface utilisateur UIX ont le support intégré pour l'internationalisation et l'accessibilité.
- Environnement de développement. On peut utiliser UIXML pour créer des pages UIX et contrôler le cours de l'application.
- Utiliser UIX pour créer des interfaces utilisateur assure un aspect ergonomique uniforme, autorisant l'utilisateur à plus se concentrer sur l'interaction homme machine.

En fournissant ces dispositifs, UIX aide à réduire la quantité de travail nécessaire pour faire tourner une application, testée, et personnalisée.

- ✓ **Positionnement par rapport aux autres langages de plasticité des interfaces :**

UIX ne génère que des pages HTML et des pages de serveur Java (JSP).
Il supporte également la génération pour les appareils mobiles.

[UIX Developer's Guide, Oracle, http://download-east.oracle.com/otn_hosted_doc/jdeveloper/904preview/uixhelp/uixdevguide/intro.html]

[UIX, Oracle, 2003, http://otn.oracle.com/products/ids/uix/index.html]

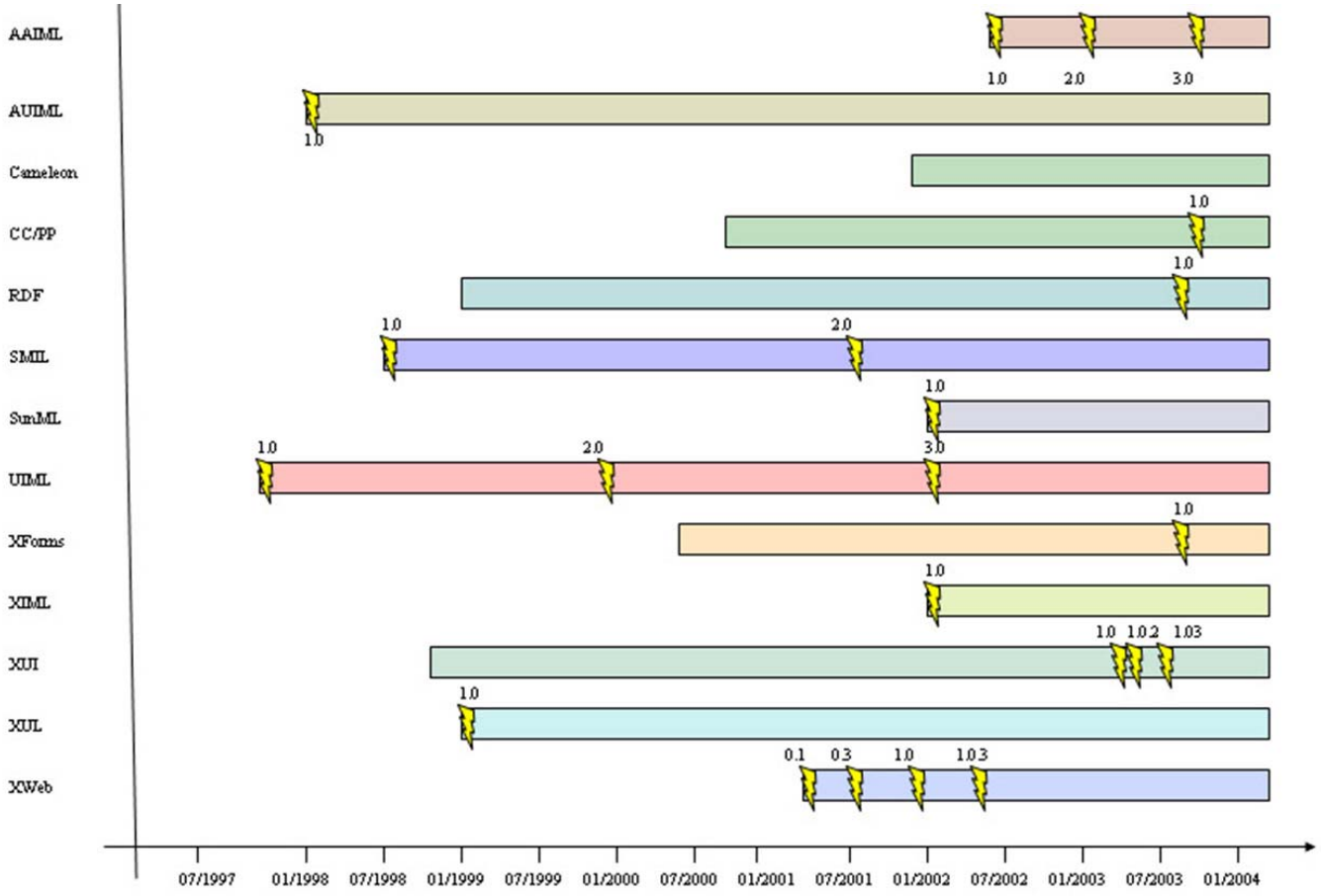
3.2. Positionnement de ces langages

Voici un résumé du positionnement des langages liés à la plasticité. On montre ici quelles sont les plateformes (PDA, WAP, Téléphone fixe ou PC) supportées par ces langages.

PLATEFORMES / LANGAGES	PDA	WAP (WML)	Mode vocal (VoiceXML)	PC
AAIML (Alternate User Interface Markup Language)	Langage suffisamment abstrait pour manipuler toutes sortes de matériels (reconnaissance vocale, radio d'une voiture ou un texte en braille)			
AUIML (Abstract User Interface Markup Language)	PalmOS			HTML, DHTML, Java Swing
CC/PP	Facilite la communication entre différents matériels			
Seescoa XML				Java Swing et AWT, HTML
SunML(Simplified Unified Natural Markup Language)				**HTML, AWT, Swing**
Teresa XML				XHTML
UIML (User Interface Markup Language)	PalmOS			HTML, Java et C++
UIX (oracle)				
XWEB				XHTML
XIML (Extensible Interface Markup Language)				HTML, Java
XForms				

3.3. Evolution de ces langages

Ce graphique illustre l'évolution des technologies liées à la plasticité depuis ces 7 dernières années. Chaque barre représente la durée de vie du langage et chaque éclair montre une étape dans l'évolution de celui-ci.



Ce graphique montre toutes les technologies liées à la plasticité et non pas les concurrents directs de Plastic ML : CC/PP, RDF, SMIL, XForms sont des technologies gravitant autour de ce sujet.

On peut voir qu'il n'y a pas beaucoup d'évolutions ces dernières années chez les concurrents de Plastic ML (UIML, SunML). Certains projets ont été abandonnés, c'est le cas de AUIML, d'autres sont tenus secrets comme XIML.
La recherche d'informations, souvent en anglais, est assez difficile. Il faut parfois rentrer dans des consortiums pour obtenir une information.

4. Critique de l'existant : Plastic ML 1.0

4.1. Capacités des langages cibles et gestion de ceux-ci par Plastic ML 1.0

4.1.1. PC

4.1.1.1. Capacités d'XHTML

4.1.1.1.1. Texte

- Taille
- Type de police
- Gras
- Italique
- Souligné
- Alignement
- Couleur
- Lien Hyper Texte

4.1.1.1.2. Tableaux

4.1.1.1.3. Formulaires

- Champ texte (input text)
- Champ texte masqué
- Zone de texte (TextArea)
- Case à cocher
- Boutons et groupe de boutons radio
- Liste déroulante
- Champ d'image
- Champ de fichier
- Bouton
- Jeu de champs (fieldset)

4.1.1.1.4. Cadres

4.1.1.1.5. Insertion d'objets

4.1.1.1.5.1. Images

- JPG
- GIF
- BMP

4.1.1.1.5.2. Vidéos

- Média Player
- Quick Time
- Real

4.1.1.1.5.3. Sons

- Mp3
- Midi
- Wav

4.1.1.1.5.4. JavaScript

4.1.1.2. Ce qui est géré par Plastic ML

4.1.1.2.1. Gestion des formulaires

- Les formulaires sont affichés sous forme de tableaux.
- Gestion des champs texte, de la taille du composant, du maximum de caractères et d'une valeur par défaut.

- Gestion des listes déroulantes (la balise « selected » est une option de Plastic ML mais elle n'est pas implémentée), en revanche, les valeurs internes et la description des options de la liste déroulante sont gérées.
- Gestion des boutons Submit et Reset du formulaire

4.1.1.3. Ce qui n'est pas géré par Plastic ML

- Aucune gestion du texte : impossible de taper un « Hello World ».
- On ne peut pas changer la police d'écriture (taille, type de police, gras, italique, alignement, couleur…).
- Impossibilité d'utiliser du JavaScript pour contrôler les formulaires.
- Dans l'interface graphique, on ne peut pas choisir le type secret d'un input.
- Pas de cases à cocher.
- Pas de zones de texte (TextArea pour l'HTML).
- Pas de boutons radios.
- Pas de champs de recherche de fichiers : bouton parcourir.
- Pas d'images.
- Pas de liens hypertextes.
- Dans les formulaires, le nombre des colonnes est fixé à 2, donc on ne peut pas mettre 2 composants (2 boutons ou 2 zones de texte par exemple).
- On ne peut pas cacher la bordure du tableau du formulaire.

4.1.2. PDA

Le navigateur Internet de Microsoft pour PDA : *Pocket Internet Explorer* gère l'affichage du WML et de l'HTML.

4.1.3. WAP

4.1.3.1. Capacités du WML

4.1.3.1.1. Mise en forme

- Taille de caractère : balise <big> et <small>
- Mise en gras (balise <b>)
- Italique (balise <i>)
- Souligné (balise <u>)

4.1.3.1.2. Paragraphe et saut de ligne

La balise <p> détermine un paragraphe et permet l'alignement gauche, droit ou centré**.** L'attribut mode de la balise <p> possède deux valeurs : wrap et nowrap. La valeur nowrap permet de ne pas effectuer de saut à la ligne (la valeur wrap permet l'inverse et est la valeur par défaut). Attention à ne pas masquer de texte (mode=nowrap) suivant la largeur de l'écran du terminal.

4.1.3.1.3. Les formulaires (Cards)

- Champ de texte (input)
- Liste déroulante (balise <select> et <option>)
- Case à cocher (option group)
- Jeu de champs (fieldset)

4.1.3.1.4. Les liens

Ils se font grâce aux balises <a> et <anchor>.

Les principaux attributs de la balise <a> sont :

- acceskey qui permet de spécifier une touche (0 à 9) de raccourci.

- href qui spécifie le lien interne ou externe.
- class : Nom de la classe donnée à la balise à laquelle elle appartient
- title : Permet de spécifier une description du lien (identique à l'attribut alt pour une image).

4.1.3.1.5. Images

Les images doivent être de préférence au format WBMP (Wireless BitMap). Ces images sont en noir et blanc, ne sont pas compressées.

4.1.3.1.6. WML script

4.1.3.2. Ce qui est géré par Plastic ML

- Les champs input sont correctement gérés (y compris le champ password).
- Le bouton submit est géré.

4.1.3.3. Ce qui n'est pas géré par Plastic ML

- Les listes déroulantes ne sont pas bien gérées. Le parseur XSL génère bien une balise <select> mais les balises <option> ont disparu. Un champ *type* est créé dans la balise select type = « popup ». Ce champ n'existe pas dans la balise select.
- L'URL générée est incorrecte, il manque un / : *http:/trg70.univ-lille1.frverification.php.*
- Les liens ne sont pas gérés.
- Les images ne sont pas gérées.
- La mise en forme n'est pas gérée.
- Un bouton *submit* est généré à la fin du formulaire en WML automatiquement.

4.1.4. Téléphone fixe

4.1.4.1. Capacités de VoiceXML

Voici la liste des balises les plus utilisées :

- Créer des variables : <assign>
- Dire un message sans attendre une réponse : <block>
- Capturer un événement : <catch>, <error> et <noinput>, <nomatch>, <help>, <throw>
- Enumérer une liste de choix possibles : <choice>, <enumerate>, <menu>
- Spécifier des touches DTMF <dtmf>
- Faire des tests : <if>, <else>, <elseif>
- Mettre une emphase au message : capacité à mettre le ou les mots aigus ou graves, faibles ou forts : <emp>
- Fermer une session : <disconnect> et <exit>
- Exécuter une action quand un champ est rempli : <filled>
- Enregistrer un flux audio : <record>

Inconvénient majeur de VoiceXML : Les champs de type input, qui permettent de saisir n'importe quelle chaîne de caractère en HTML, ne peuvent pas toujours être utilisées en VoiceXML, car les champs input doivent être associés à une grammaire. Si les réponses sont prévisibles, on pourra les

associer à une grammaire, soit en listant tous les choix possibles à la main ou alors les générer à partir d'une base de données. Si on ne peut pas prévoir les réponses, il sera préférable de ne pas utiliser VoiceXML.

4.1.4.2. Ce qui est géré par Plastic ML

L'interface générée est compatible avec VoiceXML.
Balise form : correctement générée.
Balise field : l'attribut name et la balise prompt sont correctement générés. En sortie, le fichier VoiceXML renvoie les valeurs au serveur.

4.1.4.3. Ce qui n'est pas géré par Plastic ML

La grammaire : Plastic ML permet de générer une balise <grammar src= 'fichier.gram'/>. Le programme ne génère pas de fichier fichier.gram, il faut alors le faire à la main.
Dans le cas des listes déroulantes, la grammaire pourrait être générée dans le fichier .vxml. Par exemple, si on doit saisir dans une liste déroulante entre oui et non : <grammar type="application/x-jsgf"> oui | non </grammar>.
Etant donné qu'il n'y a aucune gestion du texte dans Plastic ML, les balises de tonalité de la voix ne peuvent pas être exploitées et ne sont d'ailleurs pas implémentées.

4.2. Bugs constatés dans le Plastic ML Toolkit version 1.0

- Lorsqu'on crée un fichier Plastic ML, on ne devrait pas avoir à cliquer sur le bouton [page] puisque c'est l'élément de base qui contiendra tous les autres éléments de ce fichier.
- L'option de sauvegarde de fichier fonctionne, mais il existe cependant un problème de notation dans la boîte de dialogue car le titre de la fenêtre est "Ouvrir" alors qu'il devrait s'appeler "Sauvegarder" (ou "Enregistrer") ou "Sauvegarder sous" selon ce que veut faire l'utilisateur.
- De plus, nous avons pu remarquer un autre problème quand on sauvegarde un fichier, si nous ne mettons pas manuellement l'extension du fichier (.xml) ce n'est pas mis automatiquement. Donc nous obtenons un fichier sans extension.
- Lorsqu'on décide de quitter le programme alors que le fichier sur lequel on travaillait a été modifié depuis sa dernière sauvegarde, on affiche une boîte de dialogue pour prévenir l'utilisateur de la situation. Quand on clique sur la croix, pour fermer cette boîte de dialogue, le programme se ferme alors qu'il ne devrait pas. Le fait de cliquer sur cette croix est synonyme d'annulation et non pas d'un "NON, je ne veux pas sauvegarder".
- Lorsqu'on affiche la boîte de dialogue de sauvegarde, on retrouve le même problème.
- Si on veut créer un nouveau fichier dans le même cas, on a la même chose avec la croix.
- De plus si on annule l'opération de sauvegarde, on effectue l'opération que l'utilisateur a désiré auparavant ("Quitter" ou "Nouveau").

- La génération d'un fichier Plastic ML ayant juste l'attribut [page] ne se produit pas, hors l'utilisateur devrait pouvoir récupérer uniquement les entêtes des langages (fichiers de base des langages).
- Dans l'option "Configuration", dans la boîte de dialogue, dès qu'on appuie sur [OK] le document sur lequel on était en train de travailler est perdu. On pourra résoudre ce problème en enregistrant ce projet dans un fichier temp avant de changer les paramètres de Plastic ML et de le relancer. On réaffichera à l'écran le projet en cours à partir de ce fichier temp (qu'on pourra effacer lorsqu'il aura été chargé).
- Dans "Configuration", la sauvegarde des nouveaux paramètres fonctionne mal. En fait, il faut, après avoir tout modifié selon son désir, fermer l'application et la relancer pour que toute modification soit prise en compte.
- Quand on choisit le type d'un *field*, le champ *grammar* doit être grisé si l'utilisateur choisit de placer un bouton (*submit* ou *reset*).

5. Propositions à apporter au projet Plastic ML

5.1. Langage

- Plastic ML devrait pouvoir gérer autre chose que des formulaires. Pour le texte, on devrait pouvoir le mettre en forme ((police, gras, souligné, italique), (aigu, fort, vitesse de parole, gestion des pauses)…), gérer les liens hypertexte.
- Gestion des cases à cocher qui sont utilisées dans HTML et WML.
- Possibilité de l'utilisation des Radio Button pour HTML et remplacement de ces Radio Button en listes déroulantes pour le WML et listes de choix pour le VoiceXML.
- Gestion des TextArea en HTML et les remplacer en champ input pour le WML.
- Gestion des images pour HTML et WML selon la taille de l'image
- Gestion du découpage des pages dans le WML : découpage intelligent grâce à l'utilisation des paragraphes (utilisé en HTML et WML) ; pour représenter le passage d'un paragraphe à un autre, en VoiceXML, on pourra utiliser les pauses.
- Gestion du rôle de l'utilisateur (ex : la secrétaire n'aura pas les mêmes droits que le directeur) : On part d'un fichier Plastic ML qui va générer par l'intermédiaire d'un fichier XSL un nouveau fichier XML pour chaque rôle.
- L'équivalent d'un *input* et d'une *liste déroulante* en HTML serait une grammaire contenant les mots attendus par le concepteur de l'interface. Les mots de cette grammaire seront soit stockés dans une base de données ou soit tapés à la main. (pas toujours possible pour les champs input. Par exemple : « Saisissez votre nom » : il est impossible de créer une grammaire contenant tous les noms possibles).

5.2. Interface graphique

- Résoudre les bugs rencontrés listés ci-dessus.
- Gestion des champs passwords implémentés dans le langage mais pas dans l'interface graphique.
- Pouvoir basculer d'un langage généré à un autre à l'aide d'onglets et permettre de sauvegarder ces différents scripts.
- Intérêt réel du bouton *page* ? Il faudrait que la balise *page* soit affichée dès la création d'une page.
- Pouvoir visualiser la partie sélectionnée par l'utilisateur (mettre un cadre autour des éléments).
- Rendre l'aspect de l'interface plus esthétique et plus professionnel.
- Mieux associer les boutons avec les éléments qu'ils représentent (couleur identique).

6. Solutions apportées : Plastic ML 2.0

6.1. La DTD

6.1.1. <PlasticML>

- *Description* : Cette balise sert de balise mère. Tout document Plastic ML doit commencer par cette balise
- *Attributs* :
  - ✓ version : le numéro de la version (« 2.0 » par défaut)
- *Balises filles* : <interface>
- *Balises mères* : Ø

6.1.2. <interface>

- *Description* : Cette balise permet de décrire une interface
- *Attributs* :
  - ✓ title : le titre de l'interface
- *Balises filles* : <block>, <form>, <output>
- *Balises mères* : <PlasticML>

6.1.3. <block>

- *Description* : Permet de présenter des informations de manière insécable et les associer à des catégories des personnes (rôle). Il permet aussi d'aligner le texte.
- *Attributs* :
  - ✓ align : sert à aligner le texte : Valeurs possibles :
    - 'left' : aligner le texte à gauche
    - 'center' : aligner le texte au centre
    - 'right' : aligner le texte à droite
  - ✓ role : indique un n° de rôle : valeur possible : entre '1' et l'infini. On peut ajouter plusieurs rôles en les concaténant avec le « ; ».
  - ✓ hierarchy : indique si les rôles sont imbriqués ou pas : plus le rôle est grand plus il a le droit de voir des informations. Valeurs possibles :
    - 'yes' : il y a hiérarchie : plus le rôle est élevé, plus la personne voit de choses.
    - 'no' : il n'y a pas hiérarchie : la personne ne voit que le block dont le rôle est le même que le sien.
  - ✓ line_return : indique s'il doit y avoir des passages à la ligne quand on change de block. Valeurs possibles :
    - 'yes' : le retour à la ligne se fait entre chaque block
    - 'no' : il n'y a pas de retour à la ligne entre chaque block
- *Balises filles* : <form>, <block>, <output>, <input>
- *Balises mères* : <interface>, <block>, <form>

6.1.4. <form>

- *Description* : Permet de déclarer un formulaire
- *Attributs* :
  - ✓ name : l'identifiant du formulaire
  - ✓ method : méthode d'envoi des données au serveur, valeurs possibles : 'post' ou 'get'

- ✓ action : le nom de la page qui analysera le formulaire
- *Balises filles* : <block>, <input>, <output>, <submit>, <reset>
- *Balises mères* : <interface>, <block>

6.1.5. <input>

- *Description* : Permet de rentrer une valeur
- *Attributs* :
  - ✓ name : l'identifiant de l'input
  - ✓ type : Façon de rentrer la valeur dans l'input. Valeurs possibles :
    - 'text' : rentrer du texte
    - 'secret' : rentrer un champ caché (password)
    - 'oneselect' : saisir une valeur parmi plusieurs
    - 'multiselect' : saisir plusieurs valeurs parmi plusieurs
  - ✓ length : spécifie la longueur du champ input lorsqu'on peut l'afficher
  - ✓ maxlength : longueur maximale de l'input
  - ✓ description : Brève description de l'input
  - ✓ vxml_grammar : nom d'un fichier .gram décrivant la grammaire de l'input
  - ✓ value : valeur par défaut de l'input
- *Balises filles* : <choice>
- *Balises mères* : <block>, <form>

6.1.6. <choice>

- *Description* : Représente un élément sélectionnable d'un input 'oneselect' ou 'multiselect'
- *Attributs* :
  - ✓ default : indique la valeur par défaut. Valeurs possibles :
    - 'true' : Indique que ce choix est déjà validé
    - 'false' : Indique que ce choix n'est pas validé
  - ✓ name : Valeur externe du choix : ce qui sera donné à l'utilisateur
  - ✓ value : nom interne au formulaire
- *Balises filles* : Ø
- *Balises mères* : <input>

6.1.7. <output>

- *Description* : Permet de représenter une information de type texte (afficher un message)
- *Attributs* : Ø
- *Balises filles* : <link>, <view>, <application>, <important0>, <important1>, <important2>, <important3>, <important4>, <important5>, <important6>, <important7>
- *Balises mères* : <interface>, <block>, <form>

6.1.8. <important0>

- *Description* : Permet de décrire du texte comme le fait <output>
- *Attributs* : Ø
- *Balises mères* : <output>

- *Balises filles* : <application>

6.1.9. <important1>

- *Description* : Permet de décrire du texte avec un premier niveau d'importance. En visuel, le texte est généré en italique et en vocal par un débit de la parole plus lent (150 mots par minute) et une tessiture de la voix augmentée (+150%).
- *Attributs* : Ø
- *Balises mères* : <output>
- *Balises filles* : <application>

6.1.10. <important2>

- *Description* : Permet de décrire du texte avec un premier niveau d'importance. En visuel, le texte est généré en souligné et en vocal par un débit de la parole plus lent (150 mots par minute) et un ton plus élevé.
- *Attributs* : Ø
- *Balises mères* : <output>
- *Balises filles* : <application>

6.1.11. <important3>

- *Description* : Permet de décrire du texte avec un premier niveau d'importance. En visuel, le texte est généré en gras et en vocal par un débit de la parole plus lent (150 mots par minute) avec un volume plus élevé.
- *Attributs* : Ø
- *Balises mères* : <output>
- *Balises filles* : <application>

6.1.12. <important4>

- *Description* : Permet de décrire du texte avec un premier niveau d'importance. En visuel, le texte est généré en italique et souligné et en vocal par un débit de la parole plus lent (150 mots par minute), une tessiture de la voix augmentée (+150%) et un ton plus élevé.
- *Attributs* : Ø
- *Balises mères* : <output>
- *Balises filles* : <application>

6.1.13. <important5>

- *Description* : Permet de décrire du texte avec un premier niveau d'importance. En visuel, le texte est généré en italique et gras et en vocal par un débit de la parole plus lent (150 mots par minute), une tessiture de la voix augmentée (+150%) et un volume plus important.
- *Attributs* : Ø
- *Balises mères* : <output>
- *Balises filles* : <application>

6.1.14. <important6>

- *Description* : Permet de décrire du texte avec un premier niveau d'importance. En visuel, le texte est généré en souligné et gras et en vocal par un débit de la parole plus lent (150 mots par minute), un volume augmenté et un ton plus élevé.
- *Attributs* : Ø
- *Balises mères* : <output>
- *Balises filles* : <application>

6.1.15. <important7>

- *Description* : Permet de décrire du texte avec un premier niveau d'importance. En visuel, le texte est généré en italique, souligné et gras et en vocal par un débit de la parole plus lent (150 mots par minute), un ton plus élevé, un volume plus important et une tessiture de la voix augmentée (+150%).
- *Attributs* : Ø
- *Balises mères* : <output>
- *Balises filles* : <application>

6.1.16. <link>

- *Description* : Permet de générer un lien hypertexte.
- *Attributs* :
  - ✓ url : indique sur quelle URL aller
- *Balises mères* : <output>
- *Balises filles* : Ø

6.1.17. <view>

- *Description* : Permet d'afficher une image.
- *Attributs* :
  - ✓ src : indique le chemin absolu ou relatif de l'image à afficher
  - ✓ alt : indique un message si l'image ne peut as être affichée
  - ✓ width : largeur de l'image
  - ✓ height : hauteur de l'image
- *Balises mères* : <output>
- *Balises filles* : Ø

6.1.18. <application>

- *Description* : Permet de sélectionner une information en dehors du document Plastic ML (Résultat d'un formulaire, Base de données,…).
- *Attributs* :
  - ✓ name : indique le nom de la variable (utilisé uniquement pour afficher une variable de formulaire)
  - ✓ type : indique le type de <application>. Il peut être de type *variable (pour récupérer des informations du formulaire)*, de type *db (pour signifier que l'information que l'on cherche provient d'une base de données)* ou de type *soap*
  - ✓ connectionString : indique la chaîne de connexion (nom de la base de données pour l'instant)

- ✓ server : indique l'adresse du serveur
- ✓ login : Nom pour se connecter à la base
- ✓ password : mot de passe pour se connecter à la base
- ✓ query : la requête SQL
- ✓ soap_ref : indique la requête soap
- ✓ hidden : indique si le résultat de la balise application doit être cachée ou pas

- *Balises mères* : <output>, <input>, <important0>, <important1>, <important2>, <important3>, <important4>,<important5>, <important6>, <important7>
- *Balises filles* : Ø

6.1.19. <submit>

- *Description* : Permet de générer un bouton de validation du formulaire
- *Attributs* :
  - ✓ name : Spécifie le texte du bouton
- *Balises mères* : <form>
- *Balises filles* : Ø

6.1.20. <reset>

- *Description* : Permet de générer un bouton d'annulation du formulaire
- *Attributs* :
  - ✓ name : Spécifie le texte du bouton
- *Balises mères* : <form>
- *Balises filles* : Ø

6.2. Les transformations

Exemples en Plastic ML qui seront expliqués par la suite après transformations avec XSLT pour le XHTML, le WML et le VXML :

Exemple 1 : Hello World !

demo1.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE plasticML SYSTEM "PlasticMLv2_0.dtd">
<plasticML version="2.0">
        <interface title="Hello World !">
                <output>
                        Hello World !
                </output>
        </interface>
</plasticML>
```

Pour débuter l'explication des transformations XSL, nous partons d'un exemple simple, le Hello World : on désire « visualiser » une donnée en output, ici le texte « Hello World ! ».

Exemple 2: Importance

demo2.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE plasticML SYSTEM "PlasticMLv2_0.dtd">
<plasticML version="2.0">
        <interface title="importance">
                <output>
                        Voici une information.
                </output>

                <output>
                        <important1> Ceci est une information importante.</important1>
                </output>

                <output>
                        <important7> Ceci est une information très importante.</important7>
                </output>
        </interface>
</plasticML>
```

Cet exemple introduit la notion d'importance dans les données output. La balise <importantN> permet de décrire du texte avec un certain niveau d'importance.

Exemple 3: Retour à la ligne

demo3.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE plasticML SYSTEM "PlasticMLv2_0.dtd">
<plasticML version="2.0">
        <interface title="retour à la ligne">
                <block line_return="yes">
                        <output>
                                Voici une information.
                        </output>
                </block>
```

```
        <block line_return="yes" align="center">
                <output>
                        <important1> Ceci est une information importante.</important1>
                </output>
        </block>

        <block line_return="yes" align="right">
                <output>
                        <important7> Ceci est une information très importante.</important7>
                </output>
        </block>

    </interface>
</plasticML>
```

Dans ce 3ème exemple, nous présentons la balise block qui a différents rôles simultanément. Un de ceux-ci est d'autoriser les passages à la ligne (en mode visuel). En VoiceXML on ne génère aucune balise équivalente et on passe à ses balises filles directement. Pour cela, on attribue la valeur « yes » à l'attribut line_return. L'attribut align désigne l'alignement (« left », « right » ou «center »). La balise block permet également de rendre son contenu insécable.

Exemple 4: Image et lien

demo4.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE plasticML SYSTEM "PlasticMLv2_0.dtd">
<plasticML version="2.0">
    <interface title="Image et lien">
        <block line_return="yes">
                <output>
                        Voici une information.
                </output>
        </block>

        <block line_return="yes">
                <output>
                        <link url="demo1.plasticml">Voici un lien vers un fichier plastic ML</link>
                </output>
        </block>

        <block line_return="yes">
                <output>
                        <view src="canard.gif" alt="Image de canard" height="100" width="100"/>
                </output>
        </block>

    </interface>
</plasticML>
```

Dans ce 4ème exemple, le concepteur de la page Plastic ML veut afficher du texte simple, un lien (balise <link> dans une balise <output>) puis une image (balise <view> dans une balise <output>).

Exemple 5: Formulaire

demo5.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE plasticML SYSTEM "PlasticMLv2_0.dtd">
<plasticML version="2.0">
    <interface title="Formulaire">
        <form name="form1" action="demo6.plasticml">
            <block line_return="yes">
                <output>
                        Votre nom :
                </output>
                <input name="nom" type="text" length="10" maxlength="14"
                vxml_grammar="grammaire.gram#nom"/>
```

```
                </block>

                <block line_return="yes">
                        <output>
                                Votre mot de passe :
                        </output>
                        <input name="pwd" type="secret" length="20" maxlength="14"/>
                </block>

                <block line_return="yes">
                        <output>Mettez un commentaire</output>
                        <input name="commentaire" type="text" length="20" maxlength="14"
                vxml_grammar="grammaire.gram#nom" value="grand texte"/>
                </block>

                <block line_return="yes">
                        <output>Choisissez une couleur parmi cette petite liste de choix</output>
                        <input name="couleur1" type="oneselect" >
                                <choice name="Jaune" value="j"/>
                                <choice name="Rouge" value="r" default="yes"/>
                                <choice name="Bleu" value="b"/>
                        </input>
                </block>

                <block line_return="yes">
                        <output>Choisissez une couleur parmi cette grande liste de choix</output>
                        <input name="couleur2" type="oneselect" >
                                <choice name="Jaune" value="j"/>
                                <choice name="Rouge" value="r" default="yes"/>
                                <choice name="Bleu" value="b"/>
                                <choice name="Orange" value="o"/>
                                <choice name="Vert" value="v"/>
                                <choice name="Violet" value="vi"/>
                        </input>
                </block>

                <block line_return="yes">
                        <output>Choisissez plusieurs couleurs parmi cette petite liste de
                choix</output>
                        <input name="couleur3" type="multiselect" >
                                <choice name="Jaune" value="j"/>
                                <choice name="Rouge" value="r" default="yes"/>
                                <choice name="Bleu" value="b"/>
                        </input>
                </block>

                <block line_return="yes">
                        <output>Choisissez plusieurs couleurs parmi cette grande liste de
                choix</output>
                        <input name="couleur4" type="multiselect" length="4">
                                <choice name="Jaune" value="j"/>
                                <choice name="Rouge" value="r"/>
                                <choice name="Bleu" value="b" default="yes"/>
                                <choice name="Orange" value="o"/>
                                <choice name="Vert" value="v"/>
                                <choice name="Violet" value="vi"/>
                        </input>
                </block>

                <submit name="valider"/>

                <reset name="annuler"/>

        </form>
    </interface>
</plasticML>
```

Le créateur de demo5.plasticml souhaite récupérer des valeurs provenant d'un formulaire. Dans celui-ci, on demande le nom, le mot de passe de l'utilisateur, un commentaire, une couleur parmi une petite liste de choix, une

couleur parmi une grande liste de choix, plusieurs couleurs parmi une petite liste de choix et plusieurs couleurs parmi une grande liste de choix. Le formulaire peut être soit annulé soit validé. Une fois le formulaire validé, les données sont envoyées à la page demo6.plasticml.

Exemple 6: Saisie des données par l'utilisateur dans un formulaire et récupération des valeurs dans la page suivante

demo6.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE plasticML SYSTEM "PlasticMLv2_0.dtd">
<plasticML version="2.0">
	<interface title="Demo 6 : Inscription des données">
		<form name="form1" action="demo6-validation.plasticml">

			<block line_return="yes">
				<output>
					Votre nom :
				</output>
				<input name="nom" type="text" length="10" maxlength="14"
vxml_grammar="grammaire.gram#nom"/>
			</block>

			<block line_return="yes">
				<output>
					Votre mot de passe :
				</output>
				<input name="pwd" type="secret" length="20" maxlength="14"/>
			</block>

			<submit name="valider"/>

		</form>
	</interface>
</plasticML>
```

Avec demo6.plasticml, le nom et le mot de passe de l'utilisateur sont exigés. Celui-ci peut juste valider le formulaire une fois rempli.

demo6-validation.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE plasticML SYSTEM "PlasticMLv2_0.dtd">
<plasticML version="2.0">
	<interface title="Demo 6 : Inscription des données">

		<block line_return="yes">
			<output>Bonjour, $nom (ceci est la première méthode pour récupérer une
valeur)</output>
		</block>

		<block line_return="yes">
			<output>Bonjour, <application name="nom" type="variable"/> (ceci est la deuxième
méthode pour récupérer une valeur) </output>
		</block>

	</interface>
</plasticML>
```

Ici on voit comment récupérer le nom de cet utilisateur de deux façons différentes:

- Grâce au préfixe $ devant « nom » (nom du champ input dans lequel a été saisi le nom de l'utilisateur). La valeur est récupérée dans le fichier start.php qui gère tout et remplace les $variables par leur valeur.
- Grâce à la balise application qui a comme attributs name=« nom » (le nom du champ input désiré) et type=« variable ». La valeur est récupérée, comme pour le $, dans le fichier start.php qui gère tout et remplace les balises application par leur valeur.

La méthode avec la balise <application> semblerait s'imposer mais utiliser le dollar offre plus de souplesse à la programmation en PlasticML : en effet, grâce à ce caractère spécial, on peut faire des requêtes SQL en utilisant cette variable.

Les valeurs des input du formulaire sont retournées dans start.php par la méthode get ou post (method est un attribut de la balise form). Par défaut l'attribut method est mis à « post ».

Exemple 7: Inscription des données

demo7.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE plasticML SYSTEM "PlasticMLv2_0.dtd">
<plasticML version="2.0">
        <interface title="Demo 7 : Inscription des données">
                <form name="form1" action="demo7-validation.plasticml">

                        <block line_return="yes">
                                <output>
                                        Votre nom :
                                </output>
                                <input name="nom" type="oneselect">
                                        <application login="plasticML" password="plastic" name="log"
type="db" server="localhost" connectionString="plasticml" query="select nom
,id_personne from personne"/>
                                </input>
                        </block>

                        <block line_return="yes">
                                <output>
                                        Votre mot de passe :
                                </output>
                                <input name="pwd" type="secret" length="20" maxlength="14"/>
                        </block>

                        <submit name="valider"/>

                </form>
        </interface>
</plasticML>
```

A l'aide de demo7.plasticml, le nom de l'utilisateur est choisi parmi une liste de choix créée grâce à la balise application et son mot de passe est exigé. Pour la balise application c'est encore dans start.php qu'on s'occupe de tout remplacer. Ici on fait une requête SQL auprès de la base de données MySQL « plasticml » sur le serveur « localhost » avec comme login « plasticML » et mot de passe « plastic » pour pouvoir se connecter à cette base. La requête effectuée on récupère tous les noms et identifiants des personnes enregistrées dans la base. L'utilisateur peut juste valider le formulaire une fois rempli.

demo7-validation.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE plasticML SYSTEM "PlasticMLv2_0.dtd">
<plasticML version="2.0">
        <interface title="Demo 7 : Validation des données">

                <output>
                        Vous vous appelez <application login="plasticML" password="plastic" name="nometu"
type="db" connectionString="plasticml" server="localhost" query="select nom from personne
where id_personne = $nom"/>
                        <application login="plasticML" password="plastic" name="nometu" type="db"
connectionString="plasticml" server="localhost" query="select prenom from personne where
id_personne = $nom"/>
                </output>

        </interface>
</plasticML>
```

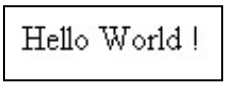
Au moyen du nom choisi dans demo6.plasticml (valeur interne de l'input, qui se trouve être l'identifiant de la personne dans la base de données) et des balises application permettant de se connecter à une base de données MySQL et d'effectuer une requête, on récupère son nom et son prénom. L'utilisateur prend connaissance de ces données.

6.2.1. XHTML

Exemple 1 : Hello World !

demo1.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
 <head>
  <title>Hello World !</title>
  <style type="text/css">
                        .souligne {text-decoration:underline}
  </style>
 </head>
 <body>
  <p>

      Hello World !

  </p>
 </body>
</html>
```

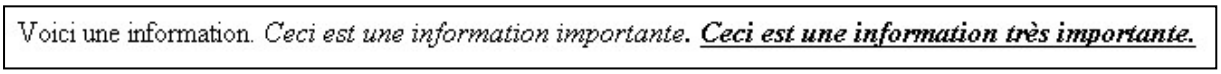


Dans cet exemple simple, on affiche juste le texte « Hello World ! ». Le reste est du code générique pour tout fichier XHTML produit.

Exemple 2: importance

demo2.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
  <head>
   <title>importance</title>
   <style type="text/css">
                        .souligne {text-decoration:underline}
   </style>
 </head>
 <body>
   <p>

      Voici une information.

      <i> Ceci est une information importante.</i>

      <i>
            <b>
                  <span class="souligne"> Ceci est une information très importante.</span>
            </b>
      </i>

   </p>
 </body>
</html>
```

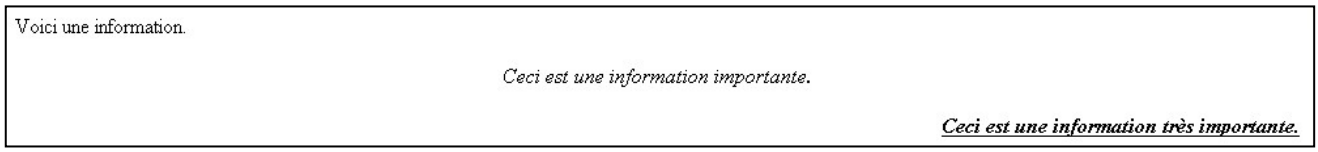


Le 1er output a été transformé en un texte simple.
Le 2ème contenant une balise important1 est transformé en un texte en italique.
Le 3ème output composé d'une balise important7 est transformé en un texte en italique, gras et souligné.

Exemple 3: retour à la ligne

demo3.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
 <head>
  <title>retour à la ligne</title>
  <style type="text/css">
                        .souligne {text-decoration:underline}
  </style>
 </head>
 <body>
  <p>

      <p>
            Voici une information.
      </p>

      <p style="text-align:center">
            <i> Ceci est une information importante.</i>
      </p>

      <p style="text-align:right">
            <i>
                  <b>
                        <span class="souligne"> Ceci est une information très importante.</span>
                  </b>
            </i>
      </p>

  </p>
 </body>
</html>
```



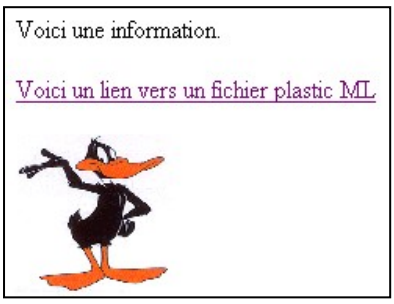
On a généré un paragraphe (<p>) pour chaque <block> dont l'attribut line_return=« yes ».
Si l'alignement était précisé dans l'attribut align, on l'a généré aussi pour le XHTML dans l'attribut style de la balise <p>. On retrouve dans cet exemple les trois manières de disposer le texte dans une page XHTML: à gauche, au milieu ou à droite.
Dans ces <p>, toutes les données contenues dans la balise <block> (dans le fichier plasticml) ont été conçues. Ici on y trouve du texte.

Exemple 4: Image et lien

demo4.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
  <head>
   <title>Image et lien</title>
   <style type="text/css">
                        .souligne {text-decoration:underline}
   </style>
 </head>
 <body>
   <p>

      <p>
            Voici une information.
      </p>
```

```
        <p>
                <a href="start.php?filenamexml=demo1.plasticml">Voici un lien vers un fichier plastic ML</a>
        </p>

        <p>
                <img src="canard.gif" alt="Image de canard" width="100" height="100"/>
        </p>

    </p>
  </body>
</html>
```

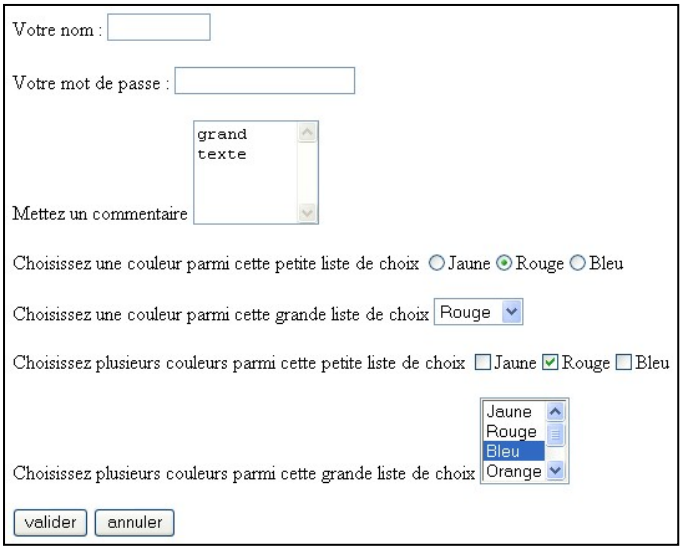


Sur cette page, on voit affiché un premier paragraphe avec du texte, ensuite un lien vers la 1ère page plasticml de démo et enfin une image de canard.

demo5.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
  <head>
   <title>Formulaire</title>
   <style type="text/css">
                        .souligne {text-decoration:underline}
   </style>
  </head>
  <body>
   <p>
        <form id="form1" method="post" action="start.php?filenamexml=demo6.plasticml">

                <p>
                        Votre nom :
                        <input type="text" name="nom" value="" size="10" maxlength="14"/>
                </p>

                <p>
                        Votre mot de passe :
                        <input type="password" name="pwd" value="" size="20" maxlength="14"/>
                </p>

                <p>
                        Mettez un commentaire
                        <textarea name="commentaire" rows="5" cols="10">grand texte</textarea>
                </p>

                <p>
                        Choisissez une couleur parmi cette petite liste de choix
                        <input type="radio" name="couleur1[]" value="j"/>Jaune
                        <input type="radio" name="couleur1[]" value="r" checked="checked"/>Rouge
                        <input type="radio" name="couleur1[]" value="b"/>Bleu
                </p>
```

```
<p>
        Choisissez une couleur parmi cette grande liste de choix
        <select name="couleur2[]">
                <option value="j">Jaune</option>
                <option value="r"
selected="selected">Rouge</option>
                <option value="b">Bleu</option>
                <option value="o">Orange</option>
                <option value="v">Vert</option>
                <option value="vi">Violet</option>
        </select>
</p>

<p>
        Choisissez plusieurs couleurs parmi cette petite liste de choix
        <input type="checkbox" name="couleur3[]" value="j"/>Jaune
         <input type="checkbox" name="couleur3[]" value="r" checked="checked"/>Rouge
         <input type="checkbox" name="couleur3[]" value="b"/>Bleu
</p>

<p>
        Choisissez plusieurs couleurs parmi cette petite liste de choix
        <select name="couleur4[]" size="4" multiple="multiple">
                <option value="j">Jaune</option>
                <option value="r">Rouge</option>
                <option value="b" selected="selected">Bleu</option>
                <option value="o">Orange</option>
                <option value="v">Vert</option>
                <option value="vi">Violet</option>
        </select>
</p>

<input type="submit" value="valider"/>

<input type="reset" value="annuler"/>

    </form>
  </p>
 </body>
</html>
```

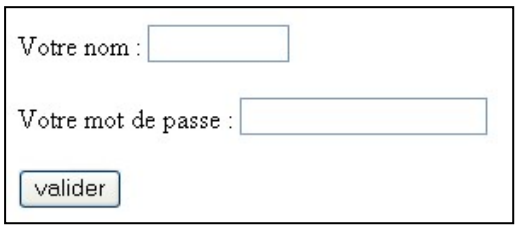


Le nom de l'utilisateur est à saisir dans une zone de saisie <input type="text"...>.
Son mot de passe est à taper dans une zone de saisie qui permet de le cacher : <input type="password"...>.
La zone du commentaire est une zone de saisie assez grande : <textarea>.
Pour le choix d'une couleur dans une petite liste, on opte pour les boutons radio : <input type="radio"…>.
Pour le choix d'une couleur dans une grande liste (plus de 5 éléments), on opte pour une liste déroulante : <select name="couleur2[]">.
Pour le choix de plusieurs couleurs dans une petite liste, on opte pour les cases à cocher : <input type="checkbox"…>.
Pour le choix de plusieurs couleurs dans une grande liste (plus de 5 éléments), on opte pour une liste dans laquelle on peut sélectionner plusieurs éléments : <select name="couleur4[]" … multiple="multiple">.

Exemple 6: Saisie des données par l'utilisateur dans un formulaire et récupération des valeurs dans la page suivante

demo6.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
 <head>
  <title>Demo 6 : Inscription des données</title>
  <style type="text/css">
                        .souligne {text-decoration:underline}
  </style>
 </head>
 <body>
  <p>
       <form id="form1" method="post" action="start.php?filenamexml=demo6-validation.plasticml">

            <p>
                  Votre nom :
                  <input type="text" name="nom" value="" size="10" maxlength="14"/>
            </p>

            <p>
                  Votre mot de passe :
                  <input type="password" name="pwd" value="" size="20" maxlength="14"/>
            </p>

            <input type="submit" value="valider"/>

       </form>
  </p>
 </body>
</html>
```



Le nom de l'utilisateur est à saisir dans une zone de saisie <input type="text"...>.
Son mot de passe est à taper dans une zone de saisie qui permet de le cacher : <input type="password"...>.

demo6-validation.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
 <head>
  <title>Demo 6 : Inscription des données</title>
  <style type="text/css">
                        .souligne {text-decoration:underline}
  </style>
 </head>
 <body>
  <p>

       <p>
            Bonjour, Pauline (ceci est la première méthode pour récupérer une valeur)
       </p>

       <p>
            Bonjour, Pauline (ceci est la deuxième méthode pour récupérer une valeur)
       </p>

  </p>
 </body>
</html>
```

Bonjour, Pauline (ceci est la première méthode pour récupérer une valeur)

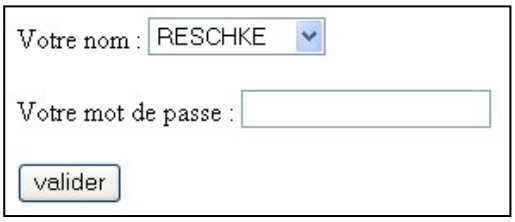
Bonjour, Pauline (ceci est la deuxième méthode pour récupérer une valeur)

Le nom saisi par l'utilisateur était « Pauline ». On voit sur cet exemple qu'avec les 2 méthodes ($ et la balise <application>) on récupère bien la même chose.

Exemple 7: Inscription des données

demo7.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
 <head>
  <title>Demo 7 : Inscription des données</title>
  <style type="text/css">
                        .souligne {text-decoration:underline}
  </style>
 </head>
 <body>
  <p>
       <form id="form1" method="post" action="start.php?filenamexml=demo7-validation.plasticml">

            <p>
                 Votre nom :
                 <input type="radio" name="nom[]" value="1"/>RESCHKE
                 <input type="radio" name="nom[]" value="2"/>CORBEAUX
                 <input type="radio" name="nom[]" value="3"/>ROUILLARD
            </p>

            <p>
                 Votre mot de passe :
                 <input type="password" name="pwd" value="" size="20" maxlength="14"/>
            </p>

            <input type="submit" value="valider"/>

       </form>
  </p>
 </body>
</html>
```



Le nom de l'utilisateur est à choisir dans une liste d'utilisateurs (ici 3 utilisateurs préexistants : 1 RESCHKE, 2 CORBEAUX et 3 ROUILLARD), liste de boutons radio <input type="radio"…>.
Son mot de passe est à taper dans une zone de saisie qui permet de le cacher : <input type="password"...>.

demo7-validation.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD Xhtml 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html>
 <head>
  <title>Demo 7 : Validation des données</title>
  <style type="text/css">
                        .souligne {text-decoration:underline}
  </style>
 </head>
 <body>
  <p>

     Vous vous appelez CORBEAUX Pauline
```

```
  </p>
 </body>
</html>
```

Vous vous appelez CORBEAUX Pauline

Le nom sélectionné par l'utilisateur était « CORBEAUX », la valeur interne de cet input était « 2 ».
Grâce à cette valeur et des balises application permettant de se connecter à une base de données MySQL et d'effectuer une requête, on récupère le nom (« CORBEAUX ») et le prénom (« Pauline ») de l'utilisateur qui a l'identifiant 2.

6.2.2. WML

Exemple 1 : Hello World !

demo1.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<wml>
	<card id="card1">
		<p>
			Hello World !
		</p>
	</card>
</wml>
```

Dans cet exemple simple, on affiche juste le texte « Hello World ! ». Le reste est du code générique pour tout fichier WML produit.

Exemple 2: importance

demo2.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<wml>
	<card id="card1">
		<p>
			Voici une information.
			<do type="accept" label="suite">
				<go href="#card2" method="post" />
			</do>
		</p>
	</card>
	<card id="card2">
		<p>

			<i>Ceci est une information importante.</i>
			<do type="accept" label="suite">
				<go href="#card3" method="post" />
			</do>
		</p>
	</card>
	<card id="card3">
		<p>
			<i>
				<b>
					<u>Ceci est une information très importante.</u>
				</b>
			</i>

		</p>
	</card>
</wml>
```

En WML, on crée un nouveau card pour chaque <output> fils de la balise <interface> (en Plastic ML). Chaque card est relié au suivant par un lien sur celui-ci.
Le 1er output a été transformé en un texte simple.
Le 2ème contenant une balise important1 est transformé en un texte en italique.
Le 3ème output composé d'une balise important7 est transformé en un texte en italique, gras et souligné.

Exemple 3: retour à la ligne

demo3.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<wml>

        <card id="card1">
                <p>
                        Voici une information.
                        <do type="accept" label="suite">
                                <go href="#card2" method="post" />
                        </do>
                </p>
        </card>

        <card id="card2">
                <p align="center">
                                <i>Ceci est une information importante.</i>
                        <do type="accept" label="suite">
                                <go href="#card3" method="post" />
                        </do>
                </p>
        </card>

        <card id="card3">
                <p align="right">
                        <i>
                                <b>
                                        <u>Ceci est une information très importante.</u>
                                </b>
                        </i>
                </p>
         </card>

</wml>
```

Ici, on crée un nouveau card pour chaque <block> fils de la balise <interface> et ne contenant que des balises <output> (en Plastic ML). Chaque card est relié au suivant par un lien sur celui-ci.
Quand l'alignement est précisé dans l'attribut align et que l'attribut line_return vaut « yes », on crée pour le WML l'attribut align de la balise <p>. On retrouve dans cet exemple les trois manières de disposer le texte dans une page WML: à gauche, au milieu ou à droite.
Dans ces <p>, toutes les données contenues dans la balise <block> (dans le fichier plasticml) ont été conçues. Ici on y trouve du texte.

Exemple 4: Image et lien

demo4.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<wml>
	<card id="card1">
		<p>

			Voici une information.

			<do type="accept" label="suite">
				<go href="#card2" method="post" />
			</do>
		</p>
	</card>
	<card id="card2">
		<p>

			<a href="start.php?filenamexml=demo1.plasticml">
				Voici un lien vers un fichier plastic ML
			</a>

			<do type="accept" label="suite">
				<go href="#card3" method="post" />
			</do>
		</p>
	</card>
	<card id="card3">
		<p>

			<img src="canard.wbmp" alt="Image de canard" width="100" height="100" />
		</p>
	</card>
</wml>
```

Sur cette page, on voit affiché un premier card avec du texte, ensuite un lien vers la 1ère page plasticml de démo et enfin une image de canard.
Sur cette page, on voit affiché un premier paragraphe avec du texte, ensuite un lien vers la 1ère page plasticml de démo et enfin une image de canard.

Exemple 5: Formulaire

demo5.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<wml>
	<card id="card1">
		<p>

			<p>
				Votre nom :
				<input type="text" name="nom" value="" size="10" maxlength="14" />
			</p>

			<do type="accept" label="suite">
				<go href="#card1-2" method="post" />
			</do>
		</p>
	</card>
	<card id="card1-2">
		<p>

			<p>
				Votre mot de passe :
				<input type="password" name="pwd" value="" size="20" maxlength="14" />
			</p>

			<do type="accept" label="suite">
				<go href="#card1-3" method="post" />
			</do>
```

```
		</p>
	</card>
	<card id="card1-3">
		<p>

			<p>
				Mettez un commentaire
				<input type="text" name="commentaire" value="grand texte" size="20"
			maxlength="14" />
			</p>

			<do type="accept" label="suite">
				<go href="#card1-4" method="post" />
			</do>
		</p>
	</card>
	<card id="card1-4">
		<p>

			<p>
				Choisissez une couleur parmi cette petite liste de choix
				<select name="couleur1" value="r;">
					<option value="j">Jaune</option>
					<option value="r">Rouge</option>
					<option value="b">Bleu</option>
				</select>
			</p>


			<do type="accept" label="suite">
				<go href="#card1-5" method="post" />
			</do>
		</p>
	</card>
	<card id="card1-5">
		<p>

			<p>
				Choisissez une couleur parmi cette grande liste de choix
				<select name="couleur2" value="r;">
					<option value="j">Jaune</option>
					<option value="r">Rouge</option>
					<option value="b">Bleu</option>
					<option value="o">Orange</option>
					<option value="v">Vert</option>
					<option value="vi">Violet</option>
				</select>
			</p>

			<do type="accept" label="suite">
				<go href="#card1-6" method="post" />
			</do>
		</p>
	</card>
	<card id="card1-6">
		<p>
			<p>
				Choisissez plusieurs couleurs parmi cette petite liste de choix
				<select name="couleur3" value="r;" multiple="true">
					<option value="j">Jaune</option>
					<option value="r">Rouge</option>
					<option value="b">Bleu</option>
				</select>
			</p>

			<do type="accept" label="suite">
				<go href="#card1-7" method="post" />
			</do>
		</p>
	</card>
	<card id="card1-7">
```

```
                <p>
                    <p>
                        Choisissez plusieurs couleurs parmi cette petite liste de choix
                        <select name="couleur4" value="b;" multiple="true">
                            <option value="j">Jaune</option>
                            <option value="r">Rouge</option>
                            <option value="b">Bleu</option>
                            <option value="o">Orange</option>
                            <option value="v">Vert</option>
                            <option value="vi">Violet</option>
                        </select>
                    </p>

                    <do type="accept" label="valider">
                        <go href="start.php?filenamexml=demo6.plasticml" method="post">
                            <postfield name="nom" value="$(nom)" />
                            <postfield name="pwd" value="$(pwd)" />
                            <postfield name="commentaire" value="$(commentaire)" />
                            <postfield name="couleur1" value="$(couleur1)" />
                            <postfield name="couleur2" value="$(couleur2)" />
                            <postfield name="couleur3" value="$(couleur3)" />
                            <postfield name="couleur4" value="$(couleur4)" />
                        </go>
                    </do>

                    <do type="reset" label="annuler">
                        <refresh>
                            <setvar name="nom" value="" />
                            <setvar name="pwd" value="" />
                            <setvar name="commentaire" value="grand texte" />
                            <setvar name="couleur1" value="r;" />
                            <setvar name="couleur2" value="r;" />
                            <setvar name="couleur3" value="r;" />
                            <setvar name="couleur4" value="b;" />
                        </refresh>
                    </do>

                </p>
            </card>
</wml>
```

Le créateur de demo5.plasticml souhaite récupérer des valeurs provenant d'un formulaire. Dans celui-ci, on demande le nom, le mot de passe de l'utilisateur, un commentaire, une couleur parmi une petite liste de choix, une couleur parmi une grande liste de choix, plusieurs couleurs parmi une petite liste de choix et plusieurs couleurs parmi une grande liste de choix. Le formulaire peut être soit annulé soit validé. Une fois le formulaire validé, les données sont envoyées à la page demo6.plasticml.
Le nom de l'utilisateur est à saisir dans une zone de saisie <input type="text"...>.
Son mot de passe est à taper dans une zone de saisie qui permet de le cacher : <input type="password"...>.
La zone du commentaire est une zone de saisie : <input type="text" …/>.
Pour le choix d'une couleur dans une petite liste, on opte pour une liste numérotée : <select name="couleur1".
Pour le choix d'une couleur dans une grande liste (plus de 5 éléments), on opte pour une liste numérotée également : <select name="couleur2">.
Pour le choix de plusieurs couleurs dans une petite liste, on opte pour des cases à cocher : <select name="couleur3" … multiple="true">.
Pour le choix de plusieurs couleurs dans une grande liste (plus de 5 éléments), on opte pour des cases à cocher également : <select name="couleur4" … multiple="true">.
En WML, pour envoyer les valeurs des champs d'un formulaire, il faut mettre tous les noms de ces champs avec leurs valeurs dans des balises <postfield> qui se trouvent dans la balise <go>, lien vers la page suivante.
Pour annuler un formulaire, tous les champs du formulaire sont mis dans des balises <setvar> avec leurs valeurs initiales, l'ensemble étant mis dans une balise <refresh> servant à rafraîchir ce formulaire.

Exemple 6: Saisie des données par l'utilisateur dans un formulaire et récupération des valeurs dans la page suivante

demo6.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<wml>
        <card id="card1">
                <p>

                        <p>
                                Votre nom :
                                <input type="text" name="nom" value="" size="10" maxlength="14" />
                        </p>

                        <do type="accept" label="suite">
                                <go href="#card1-2" method="post" />
                        </do>
                </p>
        </card>
        <card id="card1-2">
                <p>

                        <p>
                                Votre mot de passe :
                                <input type="password" name="pwd" value="" size="20" maxlength="14" />
                        </p>

                        <do type="accept" label="valider">
                                <go href="start.php?filenamexml=demo6-validation.plasticml" method="post">
                                        <postfield name="nom" value="$(nom)" />
                                        <postfield name="pwd" value="$(pwd)" />
                                </go>
                        </do>

                </p>
        </card>
</wml>
```

Le nom de l'utilisateur est à saisir dans une zone de saisie <input type="text"...>.
Son mot de passe est à taper dans une zone de saisie qui permet de le cacher : <input type="password"...>.

demo6-validation.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<wml>
        <card id="card1">
                <p>
                        Bonjour, Pauline (ceci est la première méthode pour récupérer une valeur)

                        <do type="accept" label="suite">
                                <go href="#card2" method="post" />
                        </do>
                </p>
        </card>
        <card id="card2">
                <p>
                        Bonjour, Pauline (ceci est la deuxième méthode pour récupérer une valeur)
                </p>
        </card>
</wml>
```

Le nom saisi par l'utilisateur était « Pauline ». On voit sur cet exemple qu'avec les 2 méthodes ($ et la balise <application>) on récupère bien la même chose.

Exemple 7: Inscription des données

demo7.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<wml>
        <card id="card1">
                <p>
```

```
                        <p>
                                Votre nom :
                                <select name="nom" value="">
                                        <option value="1">RESCHKE</option>
                                        <option value="2">CORBEAUX</option>
                                        <option value="3">ROUILLARD</option>
                                </select>
                        </p>

                        <do type="accept" label="suite">
                                <go href="#card1-2" method="post" />
                        </do>
                </p>
        </card>
        <card id="card1-2">
                <p>

                        <p>
                                Votre mot de passe :
                                <input type="password" name="pwd" value="" size="20" maxlength="14" />
                        </p>

                        <do type="accept" label="valider">
                                <go href="start.php?filenamexml=demo7-validation.plasticml" method="post">
                                        <postfield name="nom" value="$(nom)" /> <postfield name="pwd"
                                        value="$(pwd)" />
                                </go>
                        </do>

                </p>
        </card>
</wml>
```

Le nom de l'utilisateur est à choisir dans une liste d'utilisateurs (ici 3 utilisateurs préexistants : 1 RESCHKE, 2 CORBEAUX et 3 ROUILLARD), liste numérotée <select name="nom" …>.
Son mot de passe est à taper dans une zone de saisie qui permet de le cacher : <input type="password"...>.

demo7-validation.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<wml>
        <card id="card1">
                <p>
                        Vous vous appelez CORBEAUX Pauline

                </p>
        </card>
</wml>
```

Le nom sélectionné par l'utilisateur était « CORBEAUX », la valeur interne de cet <option> était « 2 ».
Grâce à cette valeur ainsi qu'aux balises <application> permettant de se connecter à une base de données MySQL et d'effectuer une requête, on récupère le nom (« CORBEAUX ») et le prénom (« Pauline ») de l'utilisateur qui a l'identifiant 2.

6.2.3. VoiceXML

Exemple 1 : Hello World !

demo1.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<vxml version="1.0">
        <form id="1">

                <block>
                        <prompt bargein="false">Hello World !</prompt>
                </block>

        </form>
</vxml>
```

Dans cet exemple simple, l'utilisateur entend « Hello World ! ». La balise <prompt bargein="false"> permet au serveur vocal de ne pas prendre en compte toute action de l'utilisateur, l'utilisateur ne peut pas interrompre le prompt. Le reste est du code générique pour tout fichier VoiceXML produit.

Exemple 2: importance

demo2.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<vxml version="1.0">
        <form id="1">

                <block>
                        <prompt bargein="false">Voici une information.</prompt>
                </block>

                <block>
                        <goto next="#2" />
                </block>
        </form>
        <form id="2">

                <block>
                        <prompt bargein="false">
                                <pros rate="150" range="+150%">
                                        Ceci est une information importante.
                                </pros>
                        </prompt>
                </block>

                <block>
                        <goto next="#3" />
                </block>
        </form>
        <form id="3">
                <block>
                        <prompt bargein="false">
                                <pros rate="150" pitch="100" vol="1.0" range="+150%">
                                        Ceci est une information très importante.
                                </pros>
                        </prompt>
                </block>
        </form>
</vxml>
```

Le 1er output a été transformé en un texte simple.
Pour le 2ème contenant une balise important1 et le 3ème output composé d'une balise important7, nous modulons la voix (grâce aux attributs rate (vitesse d'élocution), vol (volume), pitch (ton, hauteur de la voix) et range (tessiture de

la voix) de la balise <pros>) pour entendre la différence d'importance du texte. Pour toutes les balises <importantN>, on ralentit la vitesse d'élocution (rate="150").

Exemple 3: retour à la ligne

demo3.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<vxml version="1.0">
        <form id="1">

                <block>
                        <prompt bargein="false">Voici une information.</prompt>
                </block>

                <block>
                        <goto next="#2" />
                </block>
        </form>
        <form id="2">

                <block>
                        <prompt bargein="false">
                                <pros rate="150" range="+150%">
                                        Ceci est une information importante.
                                </pros>
                        </prompt>
                </block>

                <block>
                        <goto next="#3" />
                </block>
        </form>
        <form id="3">

                <block>
                        <prompt bargein="false">
                                <pros rate="150" pitch="100" vol="1.0" range="+150%">
                                        Ceci est une information très importante.
                                </pros>
                        </prompt>
                </block>

        </form>
</vxml>
```

Dans ce 3ème exemple, on voit que la balise <block> est totalement ignorée en VoiceXML. Le résultat de l'exemple 3 est le même que celui de l'exemple 2 où l'on ne faisait pas intervenir de balises <block>.

Exemple 4: Image et lien

demo4.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<vxml version="1.0">
        <form id="1">

                <block>
                        <prompt bargein="false">Voici une information.</prompt>
                </block>

                <block>
                        <goto next="#2" />
                </block>
        </form>
        <form id="2">
```

```
                <field>
                        <prompt>
                        Pour aller sur demo1.plasticml, dites Voici un lien vers un fichierplastic ML ou tapez 1,
                            <break time="90ms" />
                               sinon dites non.
                            <break time="200ms" />
                        </prompt>
                        <link next="start.php?filenamexml=demo1.plasticml">
                                <grammar type="application/x-jsgf">Voici un lien vers un fichier plastic
                                ML</grammar>
                                <dtmf>1</dtmf>
                        </link>
                        <grammar>non</grammar>
                </field>

                <block>
                        <goto next="#3" />
                </block>
        </form>
        <form id="3">

                <block>
                        <prompt bargein="false">
                                L'image intitulée Image de canard peutêtre visualisée en version HTML ou WML.
                        </prompt>
                </block>
        </form>
</vxml>
```

Dans ce 4ème exemple, le concepteur de la page Plastic ML veut afficher du texte simple, un lien (balise <link> dans une balise <output>) puis une image (balise <view> dans une balise <output>).
Sur cette page, l'utilisateur prend connaissance du 1er output qui est du texte. Ensuite on lui propose d'aller sur la page demo1.plasticml. Pour cela, il suffit qu'il tape 1 sur le clavier téléphonique ou qu'il dise « Voici un lien vers un fichier plastic ML ». S'il ne le désire pas, il doit dire non. Enfin, comme l'image du canard est bien évidemment non visuable sur le serveur vocal, nous informons l'utilsateur qu'il peut la voir en HTML ou WML.

Exemple 5: Formulaire

demo5.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">
        <script type="text/javascript">

                function creer_tableau(liste) {

                        var i=0;
                        var nb=0;
                        var arr = new Array();

                        while (liste.length!=0) {
                                if ((liste[i]==';') || (i==liste.length)){
                                        arr[nb++]=liste.substring(0,i);
                                        liste=liste.substring(i+1,liste.length);
                                        i=0;
                                }
                                else i++;
                        }

                        return arr;
                }


                function nom(v,ln,lv) {

                        /*création du tableau avec les noms*/
                        var arrn=creer_tableau(ln);

                        /*création du tableau avec les valeurs*/
                        var arrv=creer_tableau(lv);
```

```
                        var i=0;
                        while (arrv[i]!=v) i++;

                        return arrn[i];
                }

        </script>
        <script type="text/javascript">

                function listernoms(liste,ln,lv){

                        var result='';

                        /*création du tableau des choix de l'utilisateur*/
                        var arr=creer_tableau(liste);

                        var i=0;
                        while (i!=arr.length) {
                                if (i==0)
                                        result=nom(arr[i],ln,lv);
                                else {
                                        if (i==(arr.length-1))
                                                result=result+' et '+nom(arr[i],ln,lv);
                                        else result =result+', '+nom(arr[i],ln,lv);
                                }
                                i++;
                        }

                        return result;
                }


                function supprv(v,liste){

                        var arr = new Array();
                        var result='';

                        if  (liste) {
                                /*création du tableau des choix de l'utilisateur*/
                                arr=creer_tableau(liste);
                        }

                        var i=0;
                        while (i!=arr.length) {
                                if (arr[i]!=v)
                                        result=result+arr[i]+';';
                                        i++;
                        }

                        result=result.substring(0,result.length-1);

                        return result;

                }


                function ajoutv(v,liste,lv){

                        var i=0;
                        var arr = new Array();
                        var result='';

                        if  (liste) {
                                /*création du tableau des choix de l'utilisateur*/
                                arr=creer_tableau(liste);

                                /*on vérifie que la valeur choisie par l'utilisateur n'a pas été déjà choisie
auparavant*/
                                while (i!=arr.length) {
                                        if (arr[i]==v)
```

```
                                        break;
                                else i++;
                        }
                }

                /* on ajoute la valeur aux choix de l'utilisateur*/
                if (i==arr.length) {
                        arr[arr.length]=v;
                }

                /*création du tableau avec les valeurs*/
                var arrv=creer_tableau(lv);

                /*éléments sélectionnés mis dans l'ordre */
                i=0;
                while (i!=arrv.length) {
                        var j=0;
                        while (j!=arr.length) {
                                if (arr[j]==arrv[i])
                                        break;
                                j++;
                        }
                        if (j!=arr.length)
                                result=result+arrv[i]+';';
                        i++;
                }

                result=result.substring(0,result.length-1);

                return result;
        }

</script>
<form id="1">

        <block>
                <prompt bargein="false">
                        Votre nom :
                </prompt>
        </block>
        <field name="nom" value="">
                 <prompt></prompt>
                 <grammar src="grammaire.gram#nom" type="application/x-jsgf"/>
                 <dtmf src="grammaire.gram#nom" type="application/x-jsgf"/>
                 <filled>
                        <prompt bargein="false">Votre réponse est: <value
                        expr="nom"/>.</prompt>
                 </filled>
        </field>

        <block>
                 <prompt bargein="false">
                        Votre mot de passe :
                 </prompt>
        </block>
        <field name="pwd" value="" type="digits?maxlength=14">
                 <prompt></prompt>
                 <prompt>Par souci de confidentialité, ce mot de passe sera tapé
                 grâce aux touches de votre téléphone.</prompt>
                 <filled>
                        <prompt bargein="false">Votre réponse est: <value
                        expr="pwd"/>.</prompt>
                 </filled>
        </field>

        <block>
                <prompt bargein="false">Mettez un commentaire</prompt>
        </block>
        <field name="commentaire" value="grand texte">
                <prompt></prompt>
                <grammar src="grammaire.gram#nom" type="application/x-jsgf"/>
```

```
        <dtmf src="grammaire.gram#nom" type="application/x-jsgf"/>
        <filled>
                <prompt bargein="false">Votre réponse est: <value
                expr="commentaire"/>.</prompt>
        </filled>
</field>
```

```
<block>
        <prompt bargein="false">Choisissez une couleur parmi cette petite liste de
        choix</prompt>
</block>
<field name="couleur1">
        <prompt>Choisissez une valeur parmi les suivantes : </prompt>
        <prompt>Pour Rouge, tapez 1 ou dites Rouge.</prompt>
        <prompt>Pour Jaune, tapez 2 ou dites Jaune.</prompt>
        <prompt>Pour Bleu, tapez 3 ou dites Bleu.</prompt>
        <grammar>Jaune {j} | Rouge {r} | Bleu {b} | </grammar>
        <dtmf type="application/x-jsgf">1 {r} | 2 {j} | 3 {b} |</dtmf>
        <filled>
                <prompt bargein="false">Votre réponse est: <value
                expr="nom(couleur1,&quot;Rouge;Jaune;Bleu;&quot;,&quot;r;j;b;&quot;)
                "/>.</prompt>
        </filled>
</field>
```

```
<block>
        <prompt bargein="false">Choisissez une couleur parmi cette grande liste de
        choix</prompt>
</block>
<field name="couleur2">
        <prompt>Choisissez une valeur parmi les suivantes : </prompt>
        <prompt>Pour Rouge, tapez 1 ou dites Rouge.</prompt>
        <prompt>Pour Jaune, tapez 2 ou dites Jaune.</prompt>
        <prompt>Pour Bleu, tapez 3 ou dites Bleu.</prompt>
        <prompt>Pour Orange, tapez 4 ou dites Orange.</prompt>
        <prompt>Pour Vert, tapez 5 ou dites Vert.</prompt>
        <prompt>Pour Violet, tapez 6 ou dites Violet.</prompt>
        <grammar>Jaune {j} | Rouge {r} | Bleu {b} | Orange {o} | Vert {v} | Violet {vi}
        | </grammar>
        <dtmf type="application/x-jsgf">1 {r} | 2 {j} | 3 {b} | 4 {o} | 5 {v} | 6 {vi}
        |</dtmf>
        <filled>
                <prompt bargein="false">Votre réponse est: <value
                expr="nom(couleur2,&quot;Rouge;Jaune;Bleu;Orange;Vert;Violet;&quot;,&
                quot;r;j;b;o;v;vi;&quot;)"/>.</prompt>
        </filled>
</field>
```

```
<block>
        <prompt bargein="false">Choisissez plusieurs couleurs parmi cette petite liste de
        choix</prompt>
</block>
<block>Choix multiple</block>
<var name="listcouleur3" expr="&quot;r&quot;"/>
<field name="chxcouleur3">
        <prompt>Que voulez-vous faire?</prompt>
        <prompt>Pour ajouter un élément à votre sélection, tapez 1 ou dites
        ajouter.</prompt>
        <prompt>Pour supprimer un élément de votre sélection, tapez 2 ou dites
        supprimer.</prompt>
        <prompt>Pour consulter votre sélection actuelle, tapez 3 ou dites
        consulter.</prompt>
        <prompt>Pour valider vos choix, tapez 4 ou dites valider.</prompt>
        <grammar> ajouter {ajouter} | supprimer {supprimer} | consulter {consulter} |
        valider {valider} </grammar>
        <dtmf type="application/x-jsgf"> 1{ajouter} | 2 {supprimer} | 3 {consulter} | 4
        {valider} </dtmf>
        <filled>
                <if cond="chxcouleur3=='ajouter'">
                        <prompt bargein="false">Vous voulez ajouter un
```

```
                        élément.</prompt>
                        <clear namelist="couleur3"/>
                        <goto nextitem="couleur3"/>
                        <elseif cond="chxcouleur3=='supprimer'"/>
                        <prompt bargein="false">Vous voulez supprimer un
                        élément.</prompt>
                        <clear namelist="couleur3"/>
                        <goto nextitem="couleur3"/>
                        <elseif cond="chxcouleur3=='consulter'"/>
                        <clear namelist="chxcouleur3"/>
                        <prompt bargein="false">Votre sélection actuelle est: <value
                        expr="listernoms(listcouleur3,&quot;Rouge;Jaune;Bleu;&quot;,&q
                        uot;r;j;b;&quot;)"/>.</prompt>
                        <else/>
                        <assign name="couleur3" expr="listcouleur3"/>
                        <prompt bargein="false">Votre réponse est: <value
                        expr="&#9;listernoms(listcouleur3,&quot;Rouge;Jaune;Bleu;&quot
                        ;,&quot;r;j;b;&quot;)"/>.</prompt>
                </if>
        </filled>
</field>
<field name="couleur3">
        <prompt>Choisissez une valeur parmi les suivantes : </prompt>
        <prompt>Pour  Rouge, tapez 1 ou dites Rouge.</prompt>
        <prompt>Pour  Jaune, tapez 2 ou dites Jaune.</prompt>
        <prompt>Pour  Bleu, tapez 3 ou dites Bleu.</prompt>
        <grammar>Jaune {j} | Rouge {r} | Bleu {b} |</grammar>
        <dtmf type="application/x-jsgf">1 {r} | 2 {j} | 3 {b} |</dtmf>
        <filled>
                <if cond="chxcouleur3=='supprimer'">
                        <assign name="listcouleur3"
                        expr="supprv(couleur3,listcouleur3)"/>
                        <prompt bargein="false">L'élément <value
                        expr="nom(couleur3,&quot;Rouge;Jaune;Bleu;&quot;,&quot;r;j;b;
                        &quot;)"/> a été supprimé de votre sélection.</prompt>
                        <else/>
                        <assign name="listcouleur3"
                        expr="ajoutv(couleur3,listcouleur3,&quot;r;j;b;&quot;)"/>
                        <prompt bargein="false">L'élément <value
                        expr="nom(couleur3,&quot;Rouge;Jaune;Bleu;&quot;,&quot;r;j;b;
                        &quot;)"/> a été ajouté à votre sélection.</prompt>
                </if>
                <clear namelist="chxcouleur3"/>
        </filled>
</field>
```

```
<block>
        <prompt bargein="false">Choisissez plusieurs couleurs parmi cette petite liste de
        choix</prompt>
</block>
<block>Choix multiple</block>
<var name="listcouleur4" expr="&quot;b&quot;"/>
<field name="chxcouleur4">
        <prompt>Que voulez-vous faire?</prompt>
        <prompt>Pour ajouter un élément à votre sélection, tapez 1 ou dites
        ajouter.</prompt>
        <prompt>Pour supprimer un élément de votre sélection, tapez 2 ou dites
        supprimer.</prompt>
        <prompt>Pour consulter votre sélection actuelle, tapez 3 ou dites
        consulter.</prompt>
        <prompt>Pour valider vos choix, tapez 4 ou dites valider.</prompt>
        <grammar> ajouter {ajouter} | supprimer {supprimer} | consulter {consulter} |
        valider {valider} </grammar>
        <dtmf type="application/x-jsgf"> 1{ajouter} | 2 {supprimer} | 3 {consulter} | 4
        {valider} </dtmf>
        <filled>
                <if cond="chxcouleur4=='ajouter'">
                        <prompt bargein="false">Vous voulez ajouter un
                        élément.</prompt>
                        <clear namelist="couleur4"/>
                        <goto nextitem="couleur4"/>
```

```
                        <elseif cond="chxcouleur4=='supprimer'"/>
                        <prompt bargein="false">Vous voulez supprimer un
                        élément.</prompt>
                        <clear namelist="couleur4"/>
                        <goto nextitem="couleur4"/>
                        <elseif cond="chxcouleur4=='consulter'"/>
                        <clear namelist="chxcouleur4"/>
                        <prompt bargein="false">Votre sélection actuelle est: <value
                        expr="listernoms(listcouleur4,&quot;Bleu;Jaune;Rouge;Orange;Ve
                        rt;Violet;&quot;,&quot;b;j;r;o;v;vi;&quot;)"/>.</prompt>
                        <else/>
                        <assign name="couleur4" expr="listcouleur4"/>
                        <prompt bargein="false">Votre réponse est: <value
                        expr="&#9;listernoms(listcouleur4,&quot;Bleu;Jaune;Rouge;Orang
                        e;Vert;Violet;&quot;,&quot;b;j;r;o;v;vi;&quot;)"/>.</prompt>
                </if>
        </filled>
</field>
<field name="couleur4">
        <prompt>Choisissez une valeur parmi les suivantes : </prompt>
        <prompt>Pour  Bleu, tapez 1 ou dites Bleu.</prompt>
        <prompt>Pour  Jaune, tapez 2 ou dites Jaune.</prompt>
        <prompt>Pour  Rouge, tapez 3 ou dites Rouge.</prompt>
        <prompt>Pour  Orange, tapez 4 ou dites Orange.</prompt>
        <prompt>Pour  Vert, tapez 5 ou dites Vert.</prompt>
        <prompt>Pour  Violet, tapez 6 ou dites Violet.</prompt>
        <grammar>Jaune {j} | Rouge {r} | Bleu {b} | Orange {o} | Vert {v} | Violet {vi}
        |</grammar>
        <dtmf type="application/x-jsgf">1 {b} | 2 {j} | 3 {r} | 4 {o} | 5 {v} | 6 {vi}
        |</dtmf>
        <filled>
                <if cond="chxcouleur4=='supprimer'">
                        <assign name="listcouleur4"
                        expr="supprv(couleur4,listcouleur4)"/>
                        <prompt bargein="false">L'élément <value
                        expr="nom(couleur4,&quot;Bleu;Jaune;Rouge;Orange;Vert;Violet;
                        &quot;,&quot;b;j;r;o;v;vi;&quot;)"/> a été supprimé de votre
                        sélection.</prompt>
                        <else/>
                        <assign name="listcouleur4"
                        expr="ajoutv(couleur4,listcouleur4,&quot;b;j;r;o;v;vi;&quot;)"/>
                        <prompt bargein="false">L'élément <value
                        expr="nom(couleur4,&quot;Bleu;Jaune;Rouge;Orange;Vert;Violet;
                        &quot;,&quot;b;j;r;o;v;vi;&quot;)"/> a été ajouté à votre
                        sélection.</prompt>
                </if>
                <clear namelist="chxcouleur4"/>
        </filled>
</field>
```

```
<field type="boolean" name="valider">
         <prompt>Validez-vous ce formulaire?</prompt>
         <filled>
                  <if cond="valider">
                        <submit next="start.php?filenamexml=demo6.plasticml"
                        method="post"/>
                </if>
        </filled>
</field>
```

```
<field type="boolean" name="annuler">
         <prompt>Voulez-vous annuler ce formulaire et le remplir à nouveau?</prompt>
         <filled>
                  <if cond="annuler">
                  <goto next="#8"/>
                  </if>
         </filled>
</field>
```

```
        </form>
</vxml>
```

L'utilisateur doit prononcer son nom pour remplir le 1er champ. La grammaire grammaire.gram#nom est associée à ce champ. Ainsi il faut que le nom de l'utilisateur soit dans la grammaire pour qu'il soit reconnu.
Son mot de passe est à taper avec les touches du clavier téléphonique afin qu'il reste confidentiel.
La zone du commentaire est un champ basé sur la même grammaire que le champ n°1.
Pour le choix d'une couleur dans une petite liste ou une grande liste (plus de 5 éléments), on demande à l'utilisateur de faire un choix entre différentes valeurs : il faut soit taper le numéro correspondant à la couleur voulue, soit prononcer le nom de cette couleur.
Pour le choix de plusieurs couleurs dans une petite liste ou une grande liste (plus de 5 éléments), on demande à l'utilisateur dans un 1er temps de faire un choix : soit ajouter un élément à sa sélection, soit supprimer un élément de sa sélection, soit consulter sa sélection actuelle, ou valider ses choix.
Lorsqu'il décide d'ajouter ou de supprimer un élément, on lui demande de faire un choix entre différentes valeurs : il faut soit taper le numéro correspondant à la couleur voulue, soit prononcer le nom de cette couleur. Lorsque son choix est fait, on ajoute ou supprime la valeur et on repart dans le menu principal.
A la fin du formulaire, on demande à l'utilisateur s'il veut valider le formulaire et aller au lien suivant. Si non, on passe au champ suivant (champ pour annuler le formulaire).
Pour annuler un formulaire, on demande à l'utilisateur s'il veut annuler le formulaire. Si oui, on revient au début du formulaire.

Exemple 6: Saisie des données par l'utilisateur dans un formulaire et récupération des valeurs dans la page suivante

demo6.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<vxml version="1.0">
        <form id="1">

                <block>
                        <prompt bargein="false">Votre nom :</prompt>
                </block>
                <field name="nom" value="">
                        <prompt />
                        <grammar src="grammaire.gram#nom" type="application/x-jsgf"/>
                        <dtmf src="grammaire.gram#nom" type="application/x-jsgf" />
                        <filled>
                                <prompt bargein="false">
                                        Votre réponse est: <value expr="nom" />.
                                </prompt>
                        </filled>
                </field>

                <block>
                        <prompt bargein="false">Votre mot de passe :</prompt>
                </block>
                <field name="pwd" value="" type="digits?maxlength=14">
                        <prompt />
                        <prompt>Par souci de confidentialité, ce mot de passe sera tapégrâce aux touches de
                        votre téléphone.</prompt>
                        <filled>
                                <prompt bargein="false">
                                        Votre réponse est: <value expr="pwd" />.
                                </prompt>
                        </filled>
                </field>

                <field type="boolean" name="valider">
                        <prompt>Validez-vous ce formulaire?</prompt>
                        <filled>
                                <if cond="valider">
                                        <submit next="start.php?filenamexml=demo6-validation.plasticml"
                                        method="post" />
                                </if>
                        </filled>
                </field>
```

```
        </form>
</vxml>
```

L'utilisateur doit prononcer son nom pour remplir le 1er champ. La grammaire grammaire.gram#nom est associée à ce champ. Ainsi il faut que le nom de l'utilisateur soit dans la grammaire pour qu'il soit reconnu.
Son mot de passe est à taper avec les touches du clavier téléphonique afin qu'il reste confidentiel.

demo6-validation.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<vxml version="1.0">
        <form id="1">
                <block>
                        <prompt bargein="false">Bonjour, Pauline (ceci est la première méthode pour
                        récupérer une valeur)</prompt>
                </block>
                <block>
                        <goto next="#2" />
                </block>
        </form>
        <form id="2">
                <block>
                        <prompt bargein="false">Bonjour, Pauline (ceci est la deuxième méthode pour
                        récupérer une valeur)</prompt>
                </block>
        </form>
</vxml>
```

Le nom de l'utilisateur était « Pauline ». On voit sur cet exemple qu'avec les 2 méthodes ($ et la balise <application>) on récupère bien la même chose.

Exemple 7: Inscription des données

demo7.plasticml

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">
      <script type="text/javascript">

        function creer_tableau(liste) {

                var i=0;
                var nb=0;
                var arr = new Array();

                while (liste.length!=0) {
                        if ((liste[i]==';') || (i==liste.length)){
                                arr[nb++]=liste.substring(0,i);
                                liste=liste.substring(i+1,liste.length);
                                i=0;
                        }
                        else i++;
                }

                return arr;
        }


        function nom(v,ln,lv) {

                /*création du tableau avec les noms*/
                var arrn=creer_tableau(ln);

                /*création du tableau avec les valeurs*/
                var arrv=creer_tableau(lv);

                var i=0;
                while (arrv[i]!=v) i++;
```

```
                return arrn[i];
        }

    </script>
    <form id="1">
            <block>
                <prompt bargein="false">
                        Votre nom :
                </prompt>
            </block>
            <field name="nom">
                <prompt>Choisissez une valeur parmi les suivantes : </prompt>
                <prompt>Pour RESCHKE, tapez 1 ou dites RESCHKE.</prompt>
                <prompt>Pour CORBEAUX, tapez 2 ou dites CORBEAUX.</prompt>
                <prompt>Pour ROUILLARD, tapez 3 ou dites ROUILLARD.</prompt>
                <grammar>RESCHKE {1} | CORBEAUX {2} | ROUILLARD {3} | </grammar>
                <dtmf type="application/x-jsgf">1 {1} | 2 {2} |    3 {3} | </dtmf>
                <filled>
                        <prompt bargein="false">Votre réponse est: <value
                        expr="nom(nom,&quot;RESCHKE;CORBEAUX;ROUILLARD;&quot;,&quot;1;2;3;
                        &quot;)"/>.</prompt>
                </filled>
            </field>

            <block>
                <prompt bargein="false">
                        Votre mot de passe :
                </prompt>
            </block>
            <field name="pwd" value="" type="digits?maxlength=14">
                <prompt></prompt>
                <prompt>Par souci de confidentialité, ce mot de passe sera tapé grâce aux touches de
                votre téléphone.</prompt>
                <filled>
                      <prompt bargein="false">Votre réponse est: <value expr="pwd"/>.</prompt>
                </filled>
            </field>

            <field type="boolean" name="valider">
                <prompt>Validez-vous ce formulaire?</prompt>
                <filled>
                        <if cond="valider">
                              <submit next="start.php?filenamexml=demo7-validation.plasticml"
                          method="post"/>
                         </if>
                </filled>
            </field>

        </form>
</vxml>
```

Le nom de l'utilisateur est à choisir dans une liste d'utilisateurs (ici 3 utilisateurs préexistants : 1 RESCHKE, 2 CORBEAUX et 3 ROUILLARD). L'utilisateur peut soit taper un numéro (position de l'utilisateur dans la liste) soit dire son nom.
Son mot de passe est à taper avec les touches du clavier téléphonique afin qu'il reste confidentiel.

demo7-validation.plasticml

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<vxml version="1.0">
        <form id="1">
                <block>
                        <prompt bargein="false">Vous vous appelez CORBEAUX Pauline</prompt>
                </block>
        </form>
</vxml>
```

Le nom sélectionné par l'utilisateur était « CORBEAUX », la valeur interne dans la grammaire était « 2 ».
Grâce à cette valeur ainsi qu'aux balises <application> permettant de se connecter à une base de données MySQL et d'effectuer une requête, on récupère le nom (« CORBEAUX ») et le prénom (« Pauline ») de l'utilisateur qui a l'identifiant 2.

6.3. Le fichier start.php

Pour lancer notre application il faut taper dans un navigateur par exemple : http://localhost/plasticml/start.php?filenamexml=fichier.plasticml&lang=wml
Notre fichier PHP nous sert à exécuter les transformations XSLT en fonction de l'appareil.
Dans l'url, il faut saisir dans quel langage (variable *lang*) sera généré l'application. Si cette variable n'est pas renseignée, par défaut, de l'XHTML est généré.
Pour l'instant, la variable *lang* admet 3 valeurs : XHTML, WML ou VXML (en majuscules ou minuscules).
La variable *filenamexml* spécifie le nom du fichier plastic ML à interpréter. Si cette variable est omise, le script PHP liste tous les fichiers Plastic ML du répertoire courant et les affiche sous forme de liens.

Le traitement de la balise application est aussi géré dans le script PHP : en effet, nous avons créé un parseur XML repérant la balise *application* et établissant la connexion avec la base de données si le type est *db*. Le traitement du type *soap* se réaliserait de la même façon.
Pour tout « $variable » ou <application …> rencontré dans le fichier Plastic ML, on remplace le $variable par le contenu réel de la variable ou la balise application par la valeur de la variable, résultat de la requête SQL.

6.4. Les rôles

La notion de rôle a été intégrée à Plastic ML. Celle-ci permet d'afficher des informations réservées à une catégorie de personnes.
Pour utiliser ces rôles, on doit spécifier dans l'attribut *rôle* de la balise *application* un numéro ou plusieurs (séparés par un « ; ») :

```
<block role=  "1">
      <output> Une information </output>
</block>
```

Cette information ne sera affichée que pour une personne de rôle 1. L'identification de la personne se fait dans le script PHP. Si la personne se connecte avec un rôle égal à 2, elle ne verra rien.

La déclaration de ces rôles peuvent être hiérarchiques ou non.

- Rôles non hiérarchiques :

Supposons deux rôles : Etudiant (rôle = 1) et professeur (rôle = 2) et le code Plastic ML suivant :

```
<block role=  "1" hierarchy = "no">
      <output> Une information  pour un étudiant</output>
</block>

<block role=  "2" hierarchy = "no">
      <output> Une information  pour un professeur</output>
</block>
```

Si une personne de rôle professeur (2) se connecte, elle ne verra que les blocks de rôle 2, c'est-à-dire le deuxième block.
A noter ici que l'attribut *hierarchy* est facultatif, car, si on ne le met pas, cela veut dire qu'il n'y a pas de hiérarchie.
Lorsque les rôles ne sont pas hiérarchiques, on peut indiquer plusieurs rôles séparés par des « ; » dans l'attribut *role.*

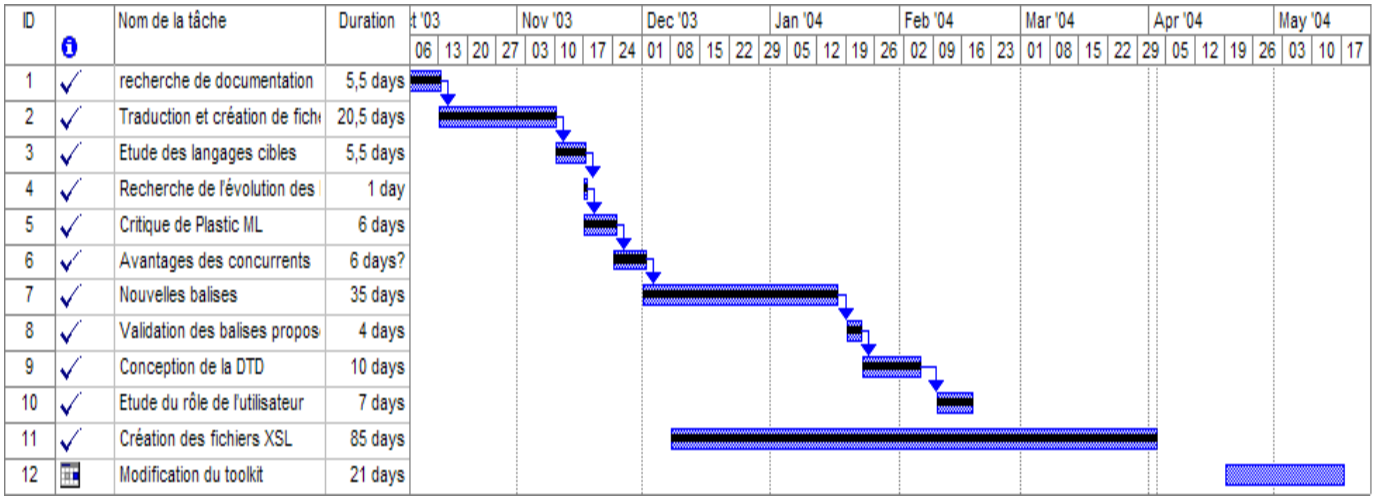
- **Rôles hiérarchiques :**

Plus la personne a un rôle élevé, plus elle a le droit de voir des informations.
Supposons deux rôles : Etudiant (rôle = 1) et professeur (rôle = 2) et le code Plastic ML suivant :

```
<block role=  "1" hierarchy = "yes">
      <output> Une information  pour un étudiant</output>
</block>

<block role=  "2" hierarchy = "yes">
      <output> Une information  pour un professeur</output>
</block>
```

Si une personne de rôle professeur (2) se connecte, elle verra les blocks de rôle 2 et aussi ceux du niveau 1, c'est-à-dire, ici, les deux blocks, alors qu’un étudiant ne verra que le block 1.

7. Planning



ID	Nom de la tâche	Duration
1	recherche de documentation	5,5 days
2	Traduction et création de fich	20,5 days
3	Etude des langages cibles	5,5 days
4	Recherche de l'évolution des	1 day
5	Critique de Plastic ML	6 days
6	Avantages des concurrents	6 days?
7	Nouvelles balises	35 days
8	Validation des balises propos	4 days
9	Conception de la DTD	10 days
10	Etude du rôle de l'utilisateur	7 days
11	Création des fichiers XSL	85 days
12	Modification du toolkit	21 days

Comme le montre notre diagramme de Gantt, nous avons au début recherché de la documentation et traduite celle-ci car elle était très souvent en anglais. Puis, après avoir étudié les langages cibles (XHTML, WML, VoiceXML), nous avons regardé comment se plaçaient les concurrents de Plastic ML (évolution et avantages). Nous avons ensuite critiqué la version 1.0 de Plastic ML en fournissant une liste exhaustive des bugs et des améliorations à fournir.
Une très grande tâche a été de constituer notre DTD, c'est-à-dire, les balises de Plastic ML. Pour cela, nous avons réalisé un brainstorming avec Mr José Rouillard. Nous avons listé toutes les balises de tous les langages cibles et nous avons enlevé les balises qui ne nous permettaient pas de générer une interface de manière abstraite et aussi certaines balises qui auraient pu poser problème dans un langage. Par exemple, les tableaux sont visualisables en XHTML et en WML, mais on ne sait pas comment les lire de manière vocale ; cette balise a donc été éliminée. En parallèle, nous avons travaillé sur les fichiers de transformations XSL qui nous ont pris 4 mois. En effet, nous avons ajouté la notion de rôle et cela a rendu notre tâche plus difficile.

8. Problèmes rencontrés

Durant ce projet, nous avons rencontré plusieurs problèmes :

- le problème majeur provient des fichiers XSL : il n'existe pas de variables en XSL. La balise <xsl : variable> sert uniquement à déclarer une constante, donc on ne peut pas modifier sa valeur. Cela nous a donc posé problème quand nous avons voulu découper le texte en WML (annulé car trop complexe), décompter les cards en WML, et passer de formulaire en formulaire en VoiceXML)
  Pour régler ce problème, nous avons dû faire appel à des template récursifs possédant des variables en paramètres.
- Problème avec VoiceXML et WML : apprentissage de ces nouveaux langages.
- En VoiceXML, intégration d'un script pour ajouter, consulter, supprimer une valeur dans les input de type oneselect et multiselect.
- Enfin, nous avons voulu mettre en ligne notre projet pour pouvoir visualiser notre application en WML. Pourquoi ? Car le simulateur de Wap, Openwave, ne supporte pas les cookies, les sessions étaient donc impossibles. Nous avons eu un problème avec l'hébergeur Free qui n'a pas la même version de PHP que nous. Free ajoute des balises de champs cachés (<input type="hidden" ...>) dans notre fichier VoiceXML. Celui-ci devient donc invalide car cette balise n'existe pas dans la DTD de VoiceXML. Il génère cette balise car Free n'envoie pas de cookie lors de la création de la session mais envoie un identifiant caché dans une balise <input>.

9. Comparaison entre Plastic ML 1.O et Plastic ML 2.O

Si l'on compare la version 1.0 de Plastic ML qui nous avait été remise au début du projet, et la 2.0, on peut voir nettement la différence :

- Nous sommes passé de 5 balises dans la version 1.0 à 20 balises dans la version 2.0.
- Nous gérons le texte, les paragraphes. De plus, ce texte peut être aligné à gauche, au centre ou à droite. Ce texte peut avoir diverses importances (7 au total) qui se représente en visuel par du texte souligné, gras, italique ou combinaison de ceux-ci et en mode vocal par diverses intonations de voix.
- Notre version permet d'afficher des images en mode visuel ou de donner une brève description de celle-ci si elle ne peut pas être vue.
- Les liens hypertextes sont gérés.
- Les listes de choix ne se résument plus à des choix uniques, on peut parfois choisir plusieurs valeurs par listes (par exemple boutons radio vs case à cocher). L'interface VoiceXML générée est très intuitive pour sélectionner plusieurs valeurs.
- On peut se connecter à une base de données pour y prendre ou y déposer des informations grâce a la balise <application> qui fait le lien avec l'extérieur. On peut aussi se servir de cette balise pour accéder à une variable (ou plutôt utiliser le $) ou même pour faire une requête SOAP (pas encore implémenté).
- Les rôles ont été intégrés. Un premier fichier XSL en amont modifie le fichier Plastic ML pour prendre en compte le rôle de l'utilisateur. Ces rôles peuvent être hiérarchiques ou non.
- Le code généré respecte :
  - la DTD XHTML 1.0 Strict
  - la DTD WML 1.1
  - la DTD VoiceXML 1.0
- On peut créer une application Plastic ML 2.0 alors que dans la version 1.0, on ne pouvait qu'afficher des formulaires incomplets sans qu'il n'y ait de traitements après (traitement des valeurs,…).

10. Ce qu'il reste à faire

Durant ce projet, nous n'avons pas eu le temps de traiter quelques aspects de Plastic ML. Celui-ci devait contenir un générateur très simple d'emploi de code Plastic ML : le Toolkit. Nous avons préféré nous concentrer sur les fichiers de transformation XSLT et offrir à ce langage une très grande grammaire.

Pour faire le lien avec l'extérieur, on n'utilise pour l'instant une connexion avec une base de données, mais l'intégration de SOAP au projet Plastic ML serait envisageable.

On a vu que le langage VoiceXML fonctionnait avec une grammaire, et lorsque l'utilisateur renseigne un champ, il faut que l'ordinateur sache à l'avance ce que la personne a le droit de rentrer. On pourrait donc générer des grammaires à partir d'une base de données par exemple.

On pourrait aussi prendre en compte d'autres langages. On a vu qu'un concurrent de Plastic ML, SunML générait du Java Swing, il faudrait peut être que Plastic ML génère du Java.

Enfin, lorsque le texte est trop long, il faudrait le découper intelligemment pour le visualiser plus facilement sur le WAP par exemple.

La gestion des tableaux pourrait être ajoutée, ainsi que d'autres sortes de présentation comme la taille ou la couleur de la police.

Conclusion

Notre projet consistait à créer la version 2.0 du langage Plastic ML. Ce langage, basé sur XML permet de décrire des interfaces Homme-Machine de manière abstraite, donc indépendantes du matériel.

Nous avons étudié les langages traitant de la plasticité des interfaces et nous nous en sommes inspirés pour ajouter de nouvelles fonctionnalités au langage.

Nous avons réalisé ce projet de la phase de veille technologique jusqu'à la phase de conception en passant par le brainstorming pour évaluer les possibilités de cette nouvelle version de ce langage.

Le langage Plastic ML a vraiment évolué, il compte maintenant 20 balises et la création d'applications Plastic ML est maintenant possible.

Ce projet nous aura permis d'avoir de solides connaissances dans les technologies XML et XSLT, ainsi que dans des nouveaux langages comme VoiceXML ou WML ainsi que de connaître les spécificités de XHTML.

Comme il nous était demandés dans le projet, nous avons ajouté la notion de rôle de l'utilisateur, donc l'interface que nous générons est en phase avec le matériel utilisé et le rôle de la personne.

Nous n'avons pas eu le temps de réaliser le Toolkit car la réalisation des fichiers XSL nous a demandés beaucoup de temps. En effet, nous devions utiliser ce langage de transformation (étudié en cours d'ADO) mais celui-ci est assez pauvre car il ne permet pas de déclarer des variables ; la balise <xsl:variable> permet uniquement de créer des constantes. Nous avons donc dû créer des templates récursifs pour simuler l'incrémentation d'une <xsl:variable>. Cette phase était assez complexe. Cela nous a au moins permis de connaître toutes les facettes de la programmation XSLT.

Avec plus de recul, nous pensons que l'utilisation de l'API SAX ou DOM nous aurait permis de gagner un temps précieux pour créer les fichiers de transformations et nous aurait ainsi permis de réaliser le Toolkit.

Le bilan de notre travail est très positif : nous générons une interface compatible avec le matériel et personnalisable suivant le rôle de la personne.

Cette gestion de rôle est peut-être à approfondir, la génération d'autres langages serait aussi souhaitable.

Bibliographie et Webographie

Livres :
- Initiation à XML (Edition Eyrolles)
- Programmation en PHP4 de Leon Atkinson (éditions CampusPress)

Outil de traduction qui a servi à la compréhension des articles anglais :
- http://dico.isc.cnrs.fr/dico_html/fr/index_tr.html

Adresses Internet relatives aux langages de plasticité :
- http://giove.cnuce.cnr.it/cameleon.html
- http://iihm.imag.fr/thevenin/these/TheseDavidThevenin.pdf
- http://xui.sourceforge.net
- http://xweb.sourceforge.net
- http://download-east.oracle.com/otn_hosted_doc/jdeveloper/904preview/uixhelp/uixdevguide/intro.html
- http://otn.oracle.com/products/ids/uix/index.html

Adresses Internet relatives aux langages de programmation :
- http://www.laltruiste.com
- http://www.w3schools.com
- http://www.w3.org
- http://www.php.net

Articles présentant la plasticité :
- ARTStudio; tool for multi-target UI design, de David Thévenin
- Workflow and Mobile Devices in Open Distance Learning, de Thomas Vantroys et José Rouillard

Articles sur Plastic ML:
- Plastic ML and its toolkit, de José Rouillard
- Plastic ML, Un langage de génération d'IHM sur mesure, de José Rouillard

Annexes

Les Acronymes

Acronyme	Signification
AAIML	Alternate Abstract Interface Markup Language
AUIML	Abstract User Interface Markup Language
CC/PP	Composite Capabilities/Preferences Profiles
Cicero	
HOMER	
Mobi-D	Model Based Interface Designer
Seescoa XML	Software Engineering for Embedded Systems using a Component-Oriented Approach
SMIL	Synchronized Multimedia Integration Language
SunML	Simplified Unified Natural Markup Language
Teresa	
UIML	User Interface Markup Language
UIX	User Interface and eXecutive
USE-IT	USable Information Technology
XForms	XML Forms
XIML	eXtensible Interface Markup Language
XUI	eXtensible User Interface
XUL	XML User Interface Language
XWeb	

La DTD

```
<!ELEMENT plasticML (interface)>
<!ATTLIST plasticML
   version CDATA #REQUIRED
>
<!ELEMENT interface (block | form | output)*>
<!ATTLIST interface
   title CDATA #REQUIRED
>
<!ELEMENT block ((form | output | block)* | (input | output | block)*)>
<!ATTLIST block
   align (left | right | center) #IMPLIED
   role CDATA #IMPLIED
   hierarchy (yes | no) #IMPLIED
   line_return (yes | no) #IMPLIED
>
<!ELEMENT form ((input | output | block)*, submit, reset?)>
<!ATTLIST form
   name CDATA #REQUIRED
   method (post | get) "post"
   action CDATA #REQUIRED
>
<!ELEMENT input (choice | application)*>
<!ATTLIST input
   name CDATA #REQUIRED
   type (text | secret | oneselect | multiselect) "text"
   length CDATA #IMPLIED
   maxlength CDATA #IMPLIED
   description CDATA #IMPLIED
   vxml_grammar CDATA #IMPLIED
   value CDATA #IMPLIED
>
<!ELEMENT choice EMPTY>
<!ATTLIST choice
   default (yes | no) "no"
   name CDATA #REQUIRED
   value CDATA #REQUIRED
>
<!-- value valeur interne -->
<!ENTITY % important "important0 | important1 | important2 | important3 | important4 | important5
| important6 | important7">
<!ELEMENT important0 (#PCDATA | application)*>
<!ELEMENT important1 (#PCDATA | application)*>
<!ELEMENT important2 (#PCDATA | application)*>
<!ELEMENT important3 (#PCDATA | application)*>
<!ELEMENT important4 (#PCDATA | application)*>
<!ELEMENT important5 (#PCDATA | application)*>
<!ELEMENT important6 (#PCDATA | application)*>
<!ELEMENT important7 (#PCDATA | application)*>
<!ELEMENT output (#PCDATA | %important; | link | view | application)*>
<!ELEMENT link (#PCDATA)>
<!ATTLIST link
   url CDATA #REQUIRED
>
<!ELEMENT view EMPTY>
<!ATTLIST view
   src CDATA #REQUIRED
   alt CDATA #REQUIRED
   width CDATA #IMPLIED
   height CDATA #IMPLIED
>
<!ELEMENT application EMPTY>
<!ATTLIST application
   name CDATA #REQUIRED
   type (variable | db | soap) #REQUIRED
   connectionString CDATA #IMPLIED
   query CDATA #IMPLIED
   server CDATA #IMPLIED
   login CDATA #IMPLIED
   password CDATA #IMPLIED
   soap_ref CDATA #IMPLIED
   hidden (yes | no) "no"
>
<!ELEMENT submit EMPTY>
<!ATTLIST submit
```

```
   name CDATA #REQUIRED
>
<!ELEMENT reset EMPTY>
<!ATTLIST reset
   name CDATA #REQUIRED
>
```