EPSI - Ecole Privée des sciences informatiques

73 rue de Marseille

33000 Bordeaux

APSIDE SA BORDEAUX

23 Quai de Paludate

33800 Bordeaux

Les interfaces graphiques

Comment obtenir une application à l'interface graphique idéale ?

Vivien POIRIER

Sylvain LABASSE

Promo 2013 – Septembre 2013

Sommaire

Introduction

Ce mémoire a pour sujet les interfaces graphiques. Mais avant toute chose, il est intéressant de savoir ce qu'est une interface graphique. Chacun de nous la côtoie tous les jours, le matin en se levant, lorsqu'on éteint son réveil de Smartphone, ou bien lorsqu'on lit les news sur notre tablette ou ordinateur portable pendant le petit déjeuner. On interagit également avec elle dans la journée, lorsqu'on travaille sur son PC. Cette interface graphique est en réalité cet ensemble d'images, de pictogrammes, de textes, de formes et de couleurs agencés les uns aux autres, avec lequel on dialogue sans cesse, que ce soit avec un téléphone standard ou un Smartphone, avec une tablette, avec le GPS intégré à la voiture, voire même le tableau de bord de son véhicule dans certains cas. On dialogue également sur PC. Les interfaces graphiques sont donc partout.

Concrètement, cette interface graphique, ou Graphical User Interface en anglais (ou GUI), va permettre aux utilisateurs d'interagir avec des dispositifs électroniques par des icônes graphiques et des indicateurs visuels. Dans cette interface, les objets à manipuler à l'écran par l'utilisateur le sont par des éléments de contrôle appelés périphériques. En effet, l'utilisation de ces périphériques va permettre d'imiter la manipulation physique de ces pictogrammes. C'est le cas avec un dispositif de pointage, qui est le plus souvent une souris. L'utilisateur va pouvoir également saisir dans cette interface des caractères, que ce soit du texte ou des caractères numériques, le plus souvent au travers d'un clavier.

Cette interface graphique est le successeur de l'interface en ligne de commande. En effet, cet ensemble permet d'utiliser un système donné de manière bien plus souple et intuitive que les lignes de commande qui sont plutôt difficiles à comprendre et à manipuler, surtout pour le grand public. La création de l'interface graphique entraina d'ailleurs l'explosion du PC et sa démocratisation au grand public. Mais nous verrons ça plus en détail un peu plus bas

Bien qu'initialement, les seuls périphériques à permettre de dialoguer et interagir avec l'interface graphique soient le clavier et la souris, il en existe bien plus aujourd'hui, et il en sort de nouveaux chaque jour. On peut citer l'exemple des pavés numériques sur les ordinateurs portables, remplaçant l'usage de la souris, ou bien l'arrivée du tactile. En effet,

le tactile, arrivé avec le Smartphone, et maintenant la tablette, modifie radicalement la manière de dialoguer avec la machine. L'ensemble des caractères sont saisis à l'aide d'un clavier virtuel s'affichant pour l'occasion quand c'est nécessaire, la navigation passe par des glissements de doigts sur l'écran, ou bien par des raccourcis tactiles qui diffèrent suivant le modèle, la marque et le système d'exploitation de l'objet électronique.

Les applications, et d'autant plus l'interface graphique, se doivent de s'adapter à tous ces nouveaux engins électroniques, afin de rester la plus intuitive et facile d'utilisation, quel que soit le terminal.

De plus, avec la rapidité croissante de l'internet, ainsi que son utilisation par toujours plus de personnes, et de manière plus intense, une demande de plus en plus importante se fait ressentir. En effet, que ce soient des personnes ou des entreprises, toutes aimeraient offrir des services ou contrôler divers équipements à partir du réseau. C'est pourquoi aujourd'hui, les applications sont majoritairement développées afin d'être accessibles via navigateur, au travers de l'intranet ou d'internet. Les applications dites client lourd, par contradiction avec les applications web, et qui fonctionnent sous un système d'exploitation spécifique, sont quant à elles de plus en plus marginalisées, bien que certaines catégories d'applications, comme les applications embarquées par exemple, ne pourrons évidemment pas être accessible via le net.

Dans ce mémoire, nous nous intéresserons plus particulièrement à ces applications accessibles via navigateur, qui sont les plus répandues aujourd'hui, mais également celles qui sont adressées au grand public.

Pour ces applications, l'interface graphique côté utilisateur va alors devoir fonctionner indépendamment du matériel, et va alors agir tel un simple client recevant et envoyant des données, le serveur gérant l'ensemble des traitements.

Voyons maintenant quand l'interface graphique est apparue, et où. Il est intéressant de noter comment celle-ci a évolué, et quand sont apparus les différents périphériques.

La première interface graphique est apparue à Palo Alto, au laboratoire de recherches Xerox PARC. En 1973, Xerox conçoit l'Alto, une station de travail totalement révolutionnaire,

combinant pour la première fois une interface totalement graphique, à base de fenêtres, avec l'utilisation de la souris. Xerox ne commercialisera pas cette machine qui est en totale rupture avec le marché. Il attendra 1981 pour commercialiser le Xerox Star, qui reprend les concepts de l'Alto.

Illustration du premier Machintosh d'Apple

C'est Apple qui fera triompher cette interface de nouvelle génération avec le projet Lisa, nouveau micro-ordinateur de la marque, qui est lancé en 1978. Suite à la visite de Steve Job au Xerox PARC, en 1979, celui-ci décide de développer une interface similaire à celle de l'Alto pour son Lisa.

Celui-ci est commercialisé en 1983, mais le lancement est un échec, car son coût est jugé trop élevé pour des performances décevantes. Dévoilé en janvier 1984, le Macintosh (voir image ci-dessus) rencontrera le succès avec cette même interface graphique totalement intégrée au système d'exploitation.

Apple peut alors se différencier face aux autres constructeurs d'ordinateurs de bureau tel que Microsoft, tous alignés sur le standard IBM PC et MS/DOS.

Les différentes initiatives de l'industrie convergent vers un standard, appelé X Window. Développé par le MIT (Massachusets Institute of Technology) en 1984, il se compose d'un logiciel serveur, d'un module client et d'un protocole de communication permettant l'échange des données de l'un à l'autre. Ce standard va donner naissance à une génération

de terminaux graphiques haute résolution. Toutes les machines Unix disposeront ainsi d'un GUI et d'une ergonomie semblable.

Microsoft lance son propre projet de GUI Windows en novembre 1985, avec un retard manifeste sur Apple. En outre, la première version de Windows s'avère fonctionnellement très limitée et son ergonomie complique plus la tâche de l'utilisateur de PC qu'elle ne l'aide.

Techniquement, Windows vient s'exécuter au-dessus de MS-DOS et souffre des limitations de celui-ci au niveau de la mémoire vive.

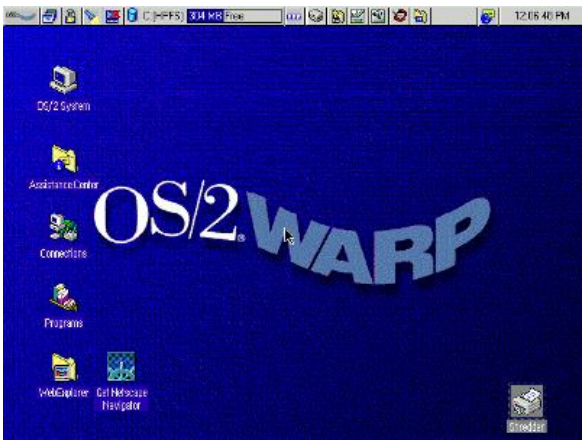

Illustration de l'interface graphique de OS /2

IBM et Microsoft coopèrent alors à la création d'un système d'exploitation de nouvelle génération : OS/2. Vendu cher, il n'apporte rien de plus que les autres produits déjà sur le marché. De plus, son interface graphique n'apparait qu'en 1988 avec sa version 1.1, dont le niveau de finition graphique est plutôt discutable.

Jusqu'en 1991, Microsoft et IBM vont faire évoluer OS/2 sans convaincre. Microsoft, dont Windows 3.0 et surtout 3.1 rencontrent enfin le succès, quitte le projet.

Débarqué de la division Macintosh en 1984 par John Sculley, PDG d'Apple Computer, Steve Jobs crée une start up. Il conçoit alors une station de travail et un système d'exploitation moderne, NextSTEP.

Dans les années 90, Apple va mal et le PDG d'alors, Gil Amelio, doit rapidement trouver un successeur à Mac OS Classic afin de replacer les ordinateurs à la pomme dans la course face à un Microsoft Windows dominateur.

Steve Jobs va être alors rappelé pour réaliser ce successeur, reprenant dans la foulée le contrôle de sa société.

NextSTEP va lui servir de base de travail pour la reconquête de ses utilisateurs. Il lance Mac OS X en 2001. IBM abandonnant son système d'exploitation OS/2, la rivalité des OS tourne au face-à-face entre Microsoft et Apple.

Le premier a assis sa domination en lançant Windows 95, puis 98, mais l'arrivée de MacOS X va démoder les interfaces utilisateurs Windows, avec des thèmes beaucoup plus élégants. Microsoft va chercher à améliorer l'apparence de Windows, maladroitement, avec Vista en 2007, puis beaucoup plus efficacement en 2009, avec Windows 7.

Une alternative à ce duopole existe, Linux. Issu d'Unix, et donc du monde des serveurs, Linux va se doter d'interfaces utilisateurs plus compatibles avec les besoins du grand public. Un grand nombre d'interfaces graphiques apparaissent dans les communautés open source pour donner au noyau Linux des interfaces performantes. KDE et Gnome s'imposent alors comme les deux interfaces les plus populaires.

Aujourd'hui, le fonctionnement d'une interface graphique fait appel à de multiples éléments qui appartiennent à diverses couches, qu'elles soient plutôt coté software, c'est-à-dire coté logiciel ou plutôt hardware, c'est-à-dire côté matériel.

La couche la plus basse, c'est-à-dire la plus proche du matériel, sera le pilote graphique, ou driver (de l'écran d'affichage) en anglais, qui va permettre au système d'exploitation de dialoguer avec l'écran.

Nous avons également l'environnement de bureau. Ce dernier correspond à un ensemble de programmes qui vont permettre de manipuler l'ordinateur à travers une interface graphique. On appelle cet ensemble de programmes l'environnement de bureau car l'interface graphique qui en découle ressemble à un bureau.

Les deux principaux environnements de bureaux sont X Window, ou X11, du fait de sa version actuelle, et Windows API, appelé également Win32. X11 est déployé sur l'ensemble des machines basées sur Linux/UNIX. Windows API est uniquement présent sur les systèmes d'exploitation de Microsoft, à partir de Windows 95.

Avant cette version, Windows était lui-même l'environnement de bureau de MS-DOS, système d'exploitation de Microsoft à l'époque. Ces environnements de bureau ont tous les deux un fonctionnement tout à fait différent, que l'on verra plus en détail par la suite.

La couche supérieure va comprendre des composants graphiques, ou widgets en anglais, qui correspondent aux objets graphiques manipulables au travers des périphériques tels que souris ou clavier.

Chaque système d'exploitation comporte au moins une bibliothèque de composants d'interface graphique par défaut, qui sera utilisée pour afficher l'ensemble des éléments de l'interface graphique. Il est également possible de surcharger ces bibliothèques graphiques par défaut afin d'y faire apparaitre d'autres éléments graphiques, avec des formes et des couleurs différentes de celles de base, accessibles via d'autres bibliothèques.

Outre la composition technique des interfaces graphiques des différents systèmes d'exploitation, les interfaces graphiques des applications que l'on développe aujourd'hui doivent être pensées et conçues afin de pouvoir être dissociées du code métier de cette application.

En effet, dans le but d'obtenir une application facilement évolutive, que ce soit au niveau des fonctionnalités, comme au niveau de l'interface graphique, la structure de la construction du code source de l'application doit respecter certaines règles et pratiques.

Ces bonnes pratiques de codage sont appelées Design Pattern en anglais, ou patrons de conception en français. Les principaux patrons sont le MVC, signifiant Modèle-Vue-Contrôleur, le MVP, signifiant Modèle-Vue-Présentateur et le MVVM, signifiant Modèle-Vue-VueModèle.

Ces différents patrons de conceptions, bien qu'ayant chacun ses spécificités et ses différences vis-à-vis des autres, vont tous les trois mettre en place une structure du code source qui va permettre de ne pas lier, de dissocier la partie relative à l'interface graphique, avec le reste du code source, composé des différents traitements que va demander l'application, appelé quant à lui code métier.

Les interfaces graphiques doivent également répondre à un certain nombre de critères. En effet, l'application doit tout d'abord proposer une interface graphique accessible au plus grand nombre, quel que soit leur profil, personnes handicapées incluses. Pour cela, il existe des bonnes pratiques à mettre en place dans le code source, comme dans les fonctionnalités de l'application, afin de répondre à cette demande.

L'application doit également répondre au critère qu'est le responsive. En effet, l'application doit être lisible et compréhensible quel que soit l'objet électronique sur lequel on affiche celle-ci, que ce soit sur PC, sur tablette, sur Smartphone, etc. Il faut donc mettre en place une interface graphique adaptable automatiquement à la résolution d'écran sur laquelle elle s'affiche. Il existe pour cela plusieurs outils qui vont aider à répondre à ce critère, tel que la solution BootStrap de Twitter.

De plus, l'ensemble de l'application doit pouvoir être contrôlée, et donc testée. Son interface graphique ne déroge pas à la règle, et doit également passer des tests, automatisés de préférence, afin de vérifier la non présence de bugs ou problèmes divers d'affichage. Divers outils et logiciels existent, et proposent de tester automatiquement l'interface en simulant un comportement utilisateur.

Pour cela, il est tout de même nécessaire que le programme en question soit paramétré pour exécuter ces tests. Il faut également que l'application propose à l'utilisateur une certaine personnalisation de son interface graphique, sans que celui ne nécessite de compétences techniques.

C'est le cas aujourd'hui quand les solutions telles que les sites web préconstruits comme Wordpress ou Drupal par exemple, proposent des thèmes, ainsi que des chartes graphiques, laissant le choix à l'utilisateur de la personnalisation de son interface. Il est enfin important

que l'application puisse permettre à des métiers non purement techniques, tels que le graphiste par exemple, modifier à l'interface simplement sans se prendre la tête, au travers de fichiers de configuration par exemple.

Il est enfin intéressant que l'ensemble des métiers qui doivent toucher à l'interface graphique, lors de la réalisation d'un projet informatique, soient identifiés. En effet, graphistes, intégrateurs, développeurs travaillent tous sur cette interface graphique.

Cependant, les impacts et les secteurs sur l'interface graphique sont tous différents. En effet, le graphiste crée le design, le développeur prépare leur utilisation dans le code, alors que l'intégrateur agence l'ensemble des travaux du graphiste dans les travaux du développeur.

Une interface graphique est donc un élément à part entière qui compose une partie non négligeable d'une application.

On peut alors se demander s'il est possible d'atteindre une certaine perfection vis-à-vis de cette interface graphique ? Ou alors comment s'y approcher au maximum ? Quelles vont être les critères d'obligations à respecter ?

Nous expliciterons dans un premier temps la structure globale d'une interface graphique, ainsi que les différents éléments qui la composent, et ce à tous les niveaux.

Nous ferons alors ressortir les similitudes et les différences de chacun des deux principales structures appelées environnements de bureau que sont Win32 de Windows et X Window des systèmes Linux/UNIX.

Puis, nous nous intéresserons aux différents objectifs auxquels l'interface graphique doit répondre aujourd'hui.

Dans un premier temps, nous verrons qu'une application doit répondre à une structure au niveau de son code afin de dissocier le fonctionnement de son interface graphique avec son code source métier.

Il nous faudra alors analyser le fonctionnement des différents patrons de conception existants allant permettre d'effectuer cette dissociation, que sont le MVC, le MVP et le MVVM.

Nous nous intéresserons alors aux différents critères qu'il faut prendre en considération concernant l'interface graphique, lorsque l'on développe un projet informatique.

Nous verrons dans un premier temps le critère d'accessibilité, qui permet à tous de consulter ou utiliser une application, même aux personnes handicapées.

Nous aurons alors l'occasion de voir le critère de Responsive, qui permet à l'application d'être lisible et consultable sans désagréments au travers de la multitude de terminaux et donc de résolutions existantes aujourd'hui.

Nous pourrons ensuite nous intéresser au critère de la testabilité de l'interface, par le biais de logiciels automatisés simulant l'utilisation de l'interface graphique par les utilisateurs. Nous verrons également comment paramétrer ces logiciels, et leur potentiel.

Nous allons alors pouvoir nous attarder sur le critère de personnalisation de l'interface, au travers la réalisation et la mise en place du système de thèmes, ou de modification de charte graphique.

Nous verrons alors les différents corps de métiers, techniques ou non, qui ont un rôle à jouer dans la réalisation de cette interface graphique, au sein de l'application.

Il y aura dans un premier temps le développeur, qui s'occupe du développement du code de l'application.

Le graphiste, quant à lui, va s'occuper de la réalisation de l'ensemble des composants graphiques nécessaires à l'interface graphique, que ce soit images, charte graphique, et autres éléments visuels.

L'intégrateur, sera le lien entre les deux autres corps de métier, et va intégrer les éléments visuels du graphiste dans le code de l'application réalisé par le développeur.

Dans la troisième et dernière partie, je m'intéresserai aux éléments que doivent intégrer ou non les différentes couches liées aux interfaces graphiques, et ce afin d'essayer de déterminer l'application avec l'interface graphique idéale.

Nous pourrons enfin conclure.

1. Structure d'une interface graphique

Nous verrons, pour chaque élément composant une interface graphique, que ce soit sous environnement Windows ou basé sur Linux, la structure de celui-ci.

1.1. Structure globale à toute interface graphique

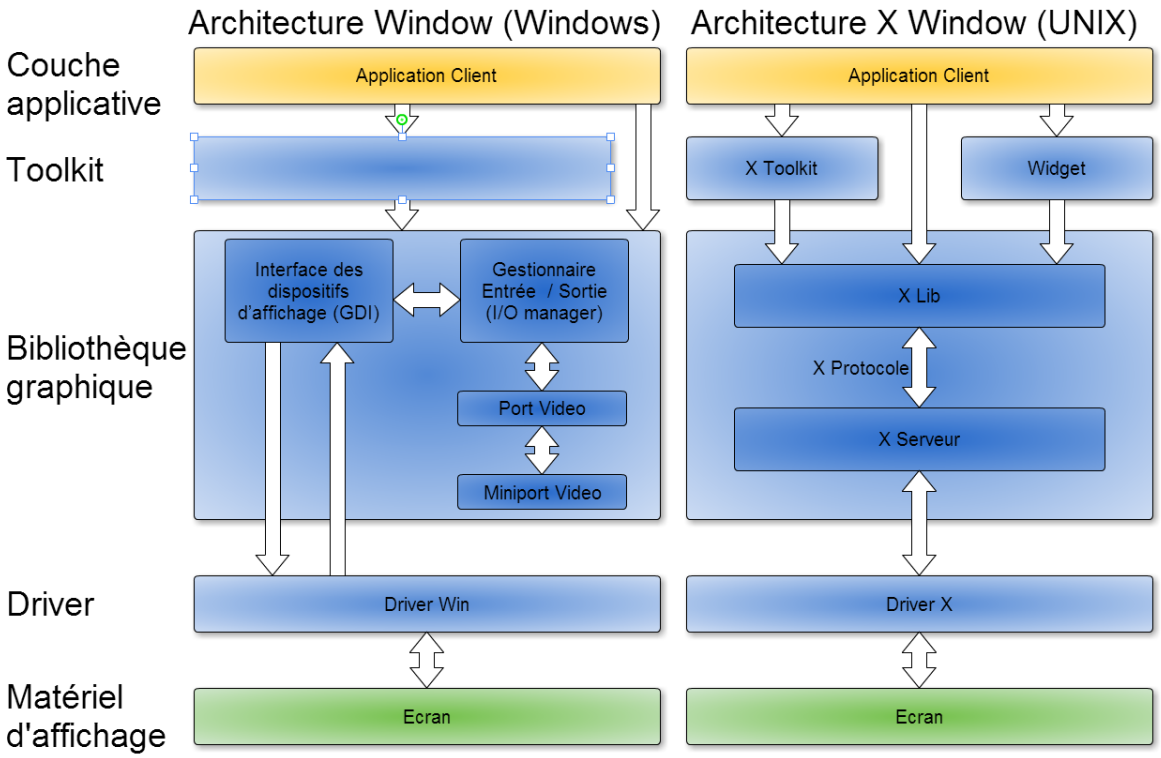

Schéma de comparaison entre les deux architectures Win32 et W Window

1.1. L'environnement de bureau

L'environnement de bureau, ou environnement graphique, ou encore desktop environment en anglais, est un ensemble de programmes qui permettent de manipuler l'ordinateur au travers d'une interface graphique. Il porte le terme d'environnement de bureau du à l'objet que représente son interface graphique, à savoir le bureau.

De nombreux systèmes d'exploitation ont un environnement de bureau incorporé. C'est le cas de Microsoft et ses systèmes d'exploitation, qui incorporent depuis Windows9X la librairie Win32, et de ce fait l'environnement de bureau qui lui est associé. Depuis Windows Vista, l'environnement de bureau se nomme Aero.

Seuls les systèmes d'exploitation basés sur Linux/UNIX, et qui contiennent donc X Window, proposent de choisir leur environnement de bureau en fonction de ceux disponibles. En pratique, il en existe un certain nombre, et les plus connus sont Gnome, KDE, Unity, et d'autres, souvent dérivés de ces environnements principaux.

En outre, il est important de faire la différence entre l'environnement de bureau et le système d'exploitation, qui sont deux éléments totalement différents. Le système d'exploitation va gérer les ressources de votre ordinateur, comme le contrôle des disques durs, des processeurs, des cartes graphiques et audio, etc., alors que l'environnement de bureau va, quant à lui, permettre d'interagir avec l'ordinateur par l'intermédiaire d'une interface graphique.

1.1.1. Boite à outils de composants / widget toolkit

Un widget toolkit, ou Boite à outils de composant d'interface graphique, est une bibliothèque logicielle destinée à concevoir des interfaces graphiques.

Cette boite à outils va contenir un ensemble de composants graphiques, chacun ayant un visuel ainsi que des propriétés qui lui sont propres. Nous verrons en détail les composants d'interface graphique dans la partie 1.2.1.

Chaque environnement de bureau en possède au moins une de base. Cependant, aujourd'hui, la plupart des applications, qu'elles soient client lourd ou accessible via navigateur, utilisent d'autres widget toolkit.

1.1.1.1. Les widget toolkit liées à un langage

De nombreuses boites à outils de composants graphiques sont disponibles sur internet.

Ceux-ci proviennent de différents framework graphiques développés par des communautés ou bien des entreprises.

Ces toolkit sont souvent intégrés aux librairies de base d'un langage, et par conséquent, deviennent les widget par défaut de ces derniers.

Par exemple, les applications client lourd développées sous Java vont utiliser le widget toolkit AWT, pour Abstract Window Toolkit. Cette bibliothèque est sortie lors de la premiere version de Java. Depuis la version deux, c'est la bibliothèque Swing qui joue le rôle de widget toolkit par défaut. AWT est toujours utilisée, bien qu'indirectement, car de nombreuses classes de Swing héritent de classes de AWT.

Le domaine d'activité où les widget toolkit ont été les plus nombreux à apparaitre est le domaine du web. En effet, avec l'explosion de l'internet, et surtout de l'internet 2.0 devenu interactif et dynamique, la plupart des applications développées aujourd'hui sont accessible via navigateur.

Ces applications sont dans la majeur partie développées à l'aide de l'architecture Java / JEE. Ces applications client – serveur ont alors un réel besoin de widget toolkit pour réaliser leur interface graphique disponible via navigateur. Face à cette demande grandissante, de nombreux framework graphiques, contenant un widget toolkit, ont été développés, et sont maintenant disponibles.

1.1.2. Notion de composant

Un composant d'interface graphique, aussi nommé widget en anglais, ou encore control par Microsoft, est un élément de base d'une interface graphique, avec lequel un utilisateur va pouvoir interagir, au travers de périphériques comme le clavier et la souris, plus récemment avec un écran tactile, etc. Ces composants sont généralement regroupés dans ce qu'on appelle des boîtes à outils de composants, comme vu plus haut. Ces composants vont alors former une interface graphique complète une fois ajoutés, classés, et agencés.

Citons les différents composants les plus utilisés.

Les composants d'interface graphique peuvent être regroupés en plusieurs catégories, en fonction de leur rôle. Nous avons les éléments d'affichage simple, les boutons, les menus, ls conteneurs, ainsi que les listes, les champs utilisateurs, les aides au retour utilisateur, ainsi que les différentes fenêtres.

1.1.2.1. Eléments d'affichage simple

Intéressons-nous tout d'abord aux éléments d'affichage simple, que sont l'étiquette et l'icône.

1.1.2.1.1. Etiquette (Label)

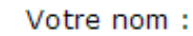

Illustration d'un label

L'étiquette, ou en anglais label, est un composant d'interface qui permet d'afficher du texte. C'est un composant statique, sans interactivité. L'étiquette est globalement utilisée pour mettre à disposition à l'utilisateur des informations ciblant souvent d'autres composants d'interface graphique situés à proximité.

1.1.2.1.2. Icône

Illustration d'une icône pour Google Chrome

L'icône se veut être un petit pictogramme voulant représenter un élément. Cela peut être une action, un objet, un logiciel, un type de fichier, etc. Les icônes servent d'outils, ayant pour objectif de rendre les interfaces utilisateur plus simples d'utilisation. Une icône est une image quelconque, comme n'importe quelle autre. Le terme « icône » employé désigne le plus souvent une image de taille relativement petite et dont le but est de représenter ou illustrer quelque chose. Un **favicon** est une utilisation d'une icône pour représenter un site web dans l'entête d'un navigateur. Il s'affiche en général sur un carré de 16 pixels de côté.

1.1.2.2. Les boutons

Intéressons-nous maintenant à la catégorie des boutons.

1.1.2.2.1. Bouton poussoir (Button)

Illustration d'un bouton poussoir

Le bouton poussoir, ou tout simplement button en anglais, est un bouton composé d'interface graphique qui fonctionne par pointer-et-cliquer. Un bouton peut soit être composé un libellé, soit une icône, soit les deux. L'ensemble devra décrire l'opération attachée au bouton.

1.1.2.2.2. Case à cocher (CheckBox)

Illustration d'une CheckBox

La case à cocher, ou check box en anglais, est un composant d'interface graphique allant permettre à l'utilisateur d'indiquer des choix. Celle-ci se compose généralement d'un libellé, et d'une case cochable / décochable. L'utilisateur peut cocher/décocher une case en cliquant dessus. Lorsqu'elle est cochée, la case est en général remplie par un "X", une croix ou une coche. Dans le cas inverse, celle-ci est laissée vide. On peut parfois observer une case grisée, ce qui arrive lorsque le choix n'est pas applicable. Il peut arriver qu'une case à cocher comporte un état intermédiaire avec une coche grisée. Ce cas de figure apparait souvent dans les cas d'arborescence, de hiérarchisation des cases à cocher. Une case avec coche grisée permet alors de signifier que certains de ses sous-éléments sont eux-mêmes cochés, et d'autres non.

1.1.2.2.3. Bouton Radio (Radio Button)

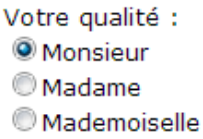

Illustration d'un bouton radio

Le bouton radio, ou radio button en anglais, est un élément d'interface graphique considéré comme un bouton qui est toujours utilisés en groupe, soit deux boutons radio au minimum. En effet, leur objectif est de permettre à l'utilisateur de choisir une seule option parmi plusieurs possibles. Un bouton radio est représenté graphiquement par un cercle et est accompagné d'un libellé qui va décrire le choix qui lui est associé. Lorsque l'utilisateur choisit une option, un point apparaît à l'intérieur du cercle pour symboliser le choix, et reste vide

dans le cas contraire. L'état initial du groupe de boutons radio peut être différent. Dans un premier cas, l'ensemble des cercles est vide et c'est à l'utilisateur de faire son choix. Dans le second cas, une option a déjà été présélectionnée. Cette deuxième situation se présente surtout lorsque l'utilisateur doit répondre à une question dont une réponse est plus courante que les autres. L'utilisateur n'a alors pas besoin de cliquer et cela permet d'aller plus vite et de gagner du temps. Dans le cas d'un groupe de boutons radio qui sont tous vides, l'utilisateur doit faire son choix en cliquant sur le cercle associé à l'option, ou si l'interface graphique le permet sur l'étiquette directement. L'utilisateur peut revenir sur son choix et cliquer sur un autre bouton radio indéfiniment, tant que le formulaire n'est pas validé. A chaque changement de choix, le point qui était présent dans le bouton radio associé à l'ancien choix disparaîtra et il réapparaîtra dans le cercle associé au nouveau choix.

1.1.2.3. Les menus

Intéressons-nous ensuite aux différents menus existants, à savoir le menu de commande, le menu contextuel et le menu circulaire.

1.1.2.3.1. Menu de commande (Command Menu)

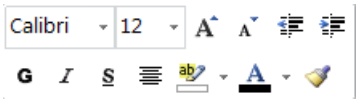

Illustration d'un menu de commande sur Word

Le menu de commande, ou Command menu en anglais, est un élément d'interface graphique généralement rectangulaire, dans lequel est présentée une liste de commandes. Les menus sont généralement cachés afin de ne pas encombrer l'espace de travail. Ils apparaissent suite à des actions précises, tels que le clic sur une zone particulière, ou sur un élément d'une barre de menu. Graphiquement, un élément de menu se compose le plus souvent d'un texte et d'un icône.

1.1.2.3.2. Menu contextuel (Context menu)

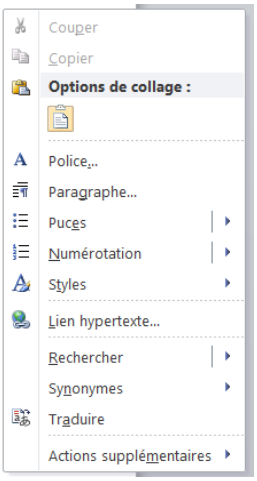

Illustration du menu contextuel de Word

Le menu contextuel, ou Context menu en anglais, désigne les menus qui s'ouvrent lorsqu'un utilisateur clique d'une façon particulière sur un objet de l'interface graphique, offrant ainsi une liste d'options qui varient selon le type de l'objet ciblé. Le plus souvent, l'utilisateur déclenche ces menus à l'aide du bouton secondaire de la souris, correspondant le plus souvent au clic droit. Cette fonctionnalité se retrouve dans les systèmes d'exploitation Microsoft Windows, Mac OS X ou UNIX combiné avec X Window System.

1.1.2.3.3. Menu circulaire (Pie menu)

Le menu circulaire, ou Pie menu en anglais, est un menu contextuel où les choix disponibles sont affichés de façon circulaire autour du curseur de la souris plutôt que d'être affichés les uns au-dessus des autres.

1.1.2.4. Les conteneurs

Voyons maintenant les différents conteneurs existants, que sont la barre d'outils, le cadre, l'onglet, et la barre de défilement.

1.1.2.4.1. Barre d'outils (Toolbar)

Illustration d'une barre d'outil sur Firefox

La barre d'outils, Toolbar en anglais, est un élément d'interfaces graphiques qui regroupe en une barre plusieurs boutons. Il s'agit donc d'une rangée d'icônes regroupées en un bloc, qui dans la plupart des logiciels, peuvent être retirées ou ajoutées de l'interface graphique.

1.1.2.4.2. Cadre (Frame)

Le cadre, ou Frame en anglais, est un type de conteneur dans lequel plusieurs composants d'interface graphique différents peuvent être regroupés. Cet effet de groupe a pour objectif de rendre l'interface graphique plus intuitive et compréhensible. Graphiquement, le cadre est représenté par une bande entourant un ensemble de composants graphiques, délimitant la zone en question.

1.1.2.4.3. L'onglet (Tab)

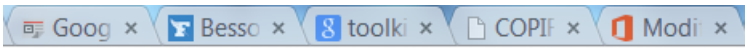

Illustration d'onglets sur Google Chrome

L'Onglet, ou Tab en anglais, est un élément d'interface graphique permettant d'avoir une interface plus riche dans une seule fenêtre. L'onglet est toujours au nombre minimum de deux. L'onglet peut être vu comme une zone de l'interface, comprenant deux éléments : une petite excroissance comprenant une étiquette toujours visible, et un ensemble de composants d'interface graphique différent. Cette deuxième partie sera visible que si l'onglet est sélectionné. Lorsqu'une interface graphique nécessite d'afficher un plus grand nombre d'éléments que ne peut afficher l'écran, l'utilisation des onglets va quand même permettre d'intégrer tous les composants. Chaque onglet comporte généralement des composants graphiques ayant un rapport les uns aux autres. Au final, dans un groupe

d’onglets, il y a toujours un seul onglet de visible. Lorsque l’utilisateur va cliquer sur l’excroissance correspondante a un onglet, le contenu existant disparait et le nouveau contenu de l’onglet choisi s’affiche.

1.1.2.4.4. Barre de défilement (Scrollbar)

Illustration de la barre de défilement

La barre de défilement, ou Scrollbar en anglais, peut également être appelée ascenseur. C’est un composant d'interface graphique liée à une zone d'affichage rectangulaire. La barre de défilement permet de faire défiler le contenu de la zone avec la souris lorsque la hauteur ou largeur de la zone est insuffisante pour afficher l'intégralité de son contenu. La barre peut être verticale pour un défilement de haut en bas ou horizontale pour un défilement de gauche à droite. Les barres se trouvent généralement sur le bord des fenêtres.

1.1.2.5. Les listes

Intéressons-nous alors à la catégorie des listes. Ces composants d’interface graphique sont la liste arborescente, la vue tabulaire, la boîte combinée et la zone de liste.

1.1.2.5.1. Liste arborescente (Tree view)

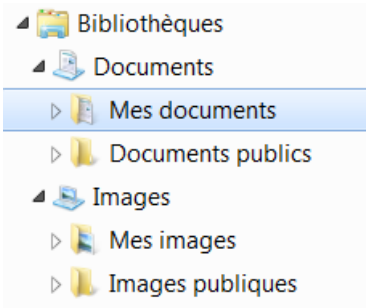

Illustration d’une liste arborescente avec l’explorateur de fichiers

La liste arborescente, ou Tree view / outline view en anglais, est un composant d'interface graphique qui modélise une vue hiérarchisée d'une information. Chaque élément, appelé nœud (node en anglais) ou branche (branch en anglais), peut avoir un certain nombre de sous éléments. Graphiquement, cette liste arborescente est représentée par une liste où chaque élément est plus ou moins indenté suivant son niveau de hiérarchie. Il est possible de plier / déplier les sous-éléments d'un élément mère, souvent à l'aide d'un double clic sur l'élément parent. Ce composant est généralement utilisé dans la navigation dans l'arborescence des différents dossiers sur PC

1.1.2.5.2. Vue tabulaire (Grid view)

Titre	Volume	Auteur	Editeur	Prix unitaire	Etat du stock	Total
20th Century Boys	1	Naoki Urasawa	Generation Comics	8,99 €	78	701,22 €
20th Century Boys	2	Naoki Urasawa	Generation Comics	8,99 €	74	665,26 €
20th Century Boys	3	Naoki Urasawa	Generation Comics	8,99 €	49	440,51 €
20th Century Boys	4	Naoki Urasawa	Generation Comics	8,99 €	47	422,53 €

Illustration d'une vue tabulaire

La Vue tabulaire ou Tableau, ou Grid view / datagrid en anglais, est un composant d'interface graphique permettant d'observer une vue tabulaire de données. Il est représenté graphiquement par un tableau rectangulaire de x lignes sur y colonnes. Il est généralement possible d'interagir avec ce tableau en pouvant trier l'affichage des données en fonction de la colonne à trier. Il est également possible de modifier la taille de chaque ligne / colonne d'un simple clic. Les données contenues dans le tableau peuvent être modifiables ou non. Le tableau est souvent appelé grid ou datagrid en anglais. Le terme data de datagrid signifie que les données proviennent directement de la base de données, et non pas celles du grid. Par exemple, dans l'explorer de fichier à partir Windows XP, la vue « Détails » correspond à une vue tabulaire.

1.1.2.5.3. Boite combinée (Combo Box)

Illustrations de la ComboBox, pliée et dépliée

La Boîte combinée, ou Combo box en anglais, est un élément d'interface graphique qui se compose d'une zone de texte et d'une liste déroulante. La liste déroulante est par défaut pliée, et affiche la valeur sélectionnée. Elle se déplie lors d'un clic souris. À l'état initial, il est possible qu'une des valeurs possibles de la liste soit affichée, ou bien que rien ne soit affiché. Ce composant est souvent couplé à un zone de saisie de texte, qui va permettre de fournir la liste des éléments sélectionnables dans la combo box.

1.1.2.5.4. Zone de liste (List Box)

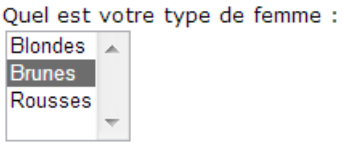

Illustration de la List Box

La zone de liste, ou List box en anglais, est un composant d'interface graphique qui permet à l'utilisateur de sélectionner un ou plusieurs éléments d'une liste. L'utilisateur doit cliquer dans la liste sur un élément pour le sélectionner. Il est possible d'en sélectionner plusieurs. Afin de désélectionner un élément, l'utilisateur à juste à recliquer sur ce dernier.

1.1.2.6. Les champs utilisateur

Voyons ensuite la catégorie des champs utilisateurs. Celle-ci va comprendre les zones de texte, les zones de mot de passe, les zones de sélection numérique et le curseur.

1.1.2.6.1. Zone de texte (Text box / Edit Field)

La Zone de texte, ou Text box ou Edit Field en anglais, est un composant d'interface graphique qui permet de proposer à l'utilisateur de saisir du texte, au travers du clavier. Graphiquement une zone de texte ressemble à un rectangle de taille quelconque à l'intérieur duquel se trouve le texte saisi par l'utilisateur. Les bordures de la zone de texte

sont en général bien mises en évidence afin de délimiter la zone de saisie. Il existe deux types de zones de texte distincts :

- les zones de texte possèdent une seule ligne de texte, qui sont appelées text box en anglais. Ces composants servent à saisir des informations concises, comme le prénom.

Illustration de la Text Box

- les zones de texte multilignes, où l'on peut rentrer un texte complet avec éventuellement des retours à la ligne, qui sont nommées text area en anglais. Celles-ci peuvent être associées à une barre de défilement lorsqu'elles ne sont pas assez grandes pour afficher l'ensemble du texte.

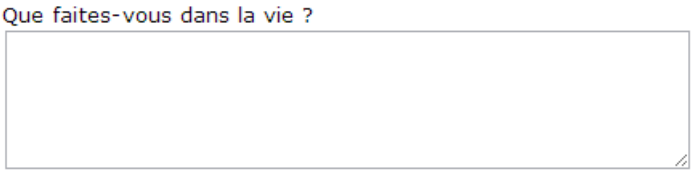

Illustration de la Text Area

Certains signaux visuels ont été intégrés afin de donner des informations à l'utilisateur sur les composants graphiques. Ici, on peut noter que le pointeur de la souris se situe sur une zone de texte, celui-ci change de forme pour indiquer à l'utilisateur qu'il peut cliquer pour que la zone de texte obtienne le focus. Une fois cette opération effectuée, un curseur se met à clignoter afin d'indiquer l'endroit où le texte sera inséré. Les zones de texte sont le plus souvent vides lorsque l'utilisateur les découvre, bien qu'aujourd'hui il ne soit pas rare de voir ces zones de textes pré remplies avec une écriture grisée, qui va alors donner des indications sur la zone de texte en question. Les zones de texte peuvent également être verrouillées en écriture pour empêcher les utilisateurs d'y modifier leur contenu. Cette possibilité permet d'indiquer à l'utilisateur que sous certaines conditions, il pourrait modifier le texte du champ. En général, il est possible de copier-coller du texte dans les zones de texte.

1.1.2.6.2. Zone de mot de passe (Password Field)

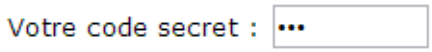

Illustration du PassWord Field

La zone de mot de passe, ou Password Field en anglais, est un moyen d'authentification pour se connecter à une ressource ou à un service dont l'accès est protégé par mot de passe. Ce champ fonctionne de la même manière que la zone de texte, bien que les caractères saisis n'apparaissent pas en clair. En effet, c'est un rond noir qui apparait pour chaque caractère saisi. De cette manière, personne ne peut lire cette donnée sensible. La seule information est le nombre de caractères saisis dans la zone de texte. Cela permet de garder la confidentialité de la chaine de caractères. Aujourd'hui, sur les périphériques mobiles, le dernier caractère saisi peut être affiché. En effet, la saisie sur petit écran tactile peut être moins précise que celle par clavier, et cette spécificité permet de confirmer ou non à l'utilisateur la saisie du bon caractère.

1.1.2.6.3. Zone de sélection numérique (Spin box)

Illustration d'une Spin Box

La zone de sélection numérique, ou Spin Box en anglais, est un composant d'interface graphique qui se compose d'une zone de texte associé à deux boutons, souvent incrustés à la zone de texte. Il n'y a pas de possibilité de saisir soit même du texte, et les seuls caractères remplissant ce composant sont numériques. La valeur est prédéterminée, et le seul moyen de la modifier est de cliquer sur les boutons, qui vont faire varier la valeur en fonction de critères prédéfinis. Cela va permettre d'ajuster une valeur à partir d'un nombre limité de valeurs possibles. De cette manière, la valeur est obligatoirement valide.

1.1.2.6.4. Curseur (Slider)

Illustration d'un Slider

Le curseur, ou Slider en anglais, mais encore appelé track bar par Microsoft, est un composant d'interface graphique qui ressemble à une droite graduée, liée à un indicateur de position. L'utilisateur peut faire glisser cet indicateur pour sélectionner la valeur qu'il souhaite. Il peut parfois également cliquer directement sur la droite à l'endroit correspondant à la valeur qu'il souhaite sélectionner. Généralement, une zone de texte est ajoutée afin d'afficher la valeur sélectionnée, et ce pour une meilleure lisibilité.

1.1.2.7. Les aides au retour utilisateur

Les aides au retour utilisateur sont une catégorie de composants d'interface graphique à ne pas négliger. Ils tiennent l'utilisateur informé de divers événements ou actions, ou bien d'éléments présents sur l'interface. Ces aides au retour utilisateur sont composées de barre de progression, de barre d'état et la bulle d'aide.

1.1.2.7.1. Barre de progression (Progress Bar)

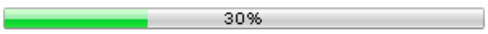

Illustration d'une progress bar

La barre de progression, ou Progress bar en anglais, est un composant d'interface graphique indiquant à l'utilisateur l'état d'avancement d'un traitement que l'ordinateur est train d'effectuer. Initialement, la barre est complètement vide, puis elle se remplit au fur et à mesure de l'avancement de la tâche pour finir complètement remplie lorsque le traitement est achevé. En plus de la représentation graphique, l'avancement est souvent également indiqué à l'aide soit d'un pourcentage, soit du nombre d'éléments traités sur le nombre

total, ou d'une autre manière. On retrouve ces barres de progression par exemple pendant les transferts de fichiers, les téléchargements sur internet, ou lors d'un scan d'antivirus. Lorsqu'il est impossible de connaître l'état d'avancement d'un travail (par exemple, un processus de téléchargement d'un fichier qui ne connaît pas encore sa taille totale), la barre de progression s'adapte à cette situation avec une signalétique particulière qui diffère.

1.1.2.7.2. Barre d'état (Status Bar)

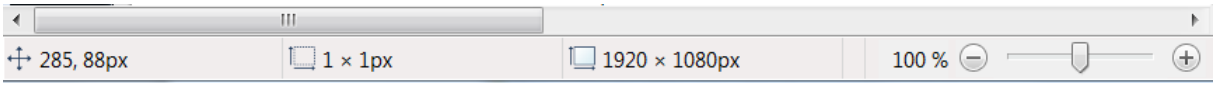

Illustration d'une barre d'état sur Paint

La barre d'état, ou Status bar en anglais, est un composant d'interface graphique qui correspond à une partie d'une fenêtre où est affiché l'état du programme associé à la fenêtre. Pour la plupart des programmes, la barre d'état se traduit par une barre horizontale qui est affichée en bas de la fenêtre. Celle-ci va comprendre un certain nombre d'informations qui sont en rapport avec le logiciel utilisé. Pour Microsoft Word par exemple, la barre d'état, située en bas de la fenêtre, contient le numéro de la page actuelle, le nombre total de pages du document, le nombre de mots, etc.

1.1.2.7.3. Bulle d'aide (Tooltip)

Illustration d'un tooltip sur word

La Bulle d'aide, ou Tooltip en anglais, est composant d'interface graphique qui a pour objectif d'aider l'utilisateur en donner une information. Graphiquement, il se traduit par un message qui apparaît lors du passage du curseur de la souris sur un élément.

1.1.2.8. Les fenêtres

Voyons enfin la catégorie des fenêtres, ou Window en anglais. Les composants d'interface graphique qui la composent sont la fenêtre simple, la fenêtre modale, la boîte de dialogue et la fenêtre flottante.

1.1.2.8.1. Fenêtre simple

La fenêtre simple, ou Window en anglais, est un élément de l'interface graphique. C'est une zone rectangulaire de l'écran qui va s'occuper de l'affichage d'une partie de logiciel, ou d'un logiciel entier. Les fenêtres sont parfois appelées feuilles, ou sheet en anglais. Le contenu de la fenêtre appartient au logiciel, alors que la bordure et le bandeau supérieur sont gérés par l'environnement graphique. C'est ce dernier qui permet le déplacement et le redimensionnement des fenêtres. Il est donc nécessaire que le programme et l'environnement soient compatibles afin d'interagir convenablement.

1.1.2.8.2. Fenêtre modale (Modal window)

La fenêtre modale, ou Modal window en anglais est un composant d'interface graphique de type fenêtre. Celle-ci prend le contrôle total du clavier et de l'écran. Elle est en général associée à une information importante que l'utilisateur doit lire et valider avant de poursuivre, ou de modifier quoi que ce soit. C'est par exemple le cas des fenêtres pop-up qui affichent un message d'erreur, que l'on doit valider, et qui nous interdit de basculer sur la fenêtre principale du logiciel.

1.1.2.8.3. Boite de dialogue (Dialog box)

La boîte de dialogue, ou Dialog box en anglais, est un composant d'interface graphique constitué d'une fenêtre qui est similaire à la fenêtre modale. Celle-ci prend également le contrôle total du clavier et de l'écran, et affiche une information, souvent suivie d'une question. L'utilisateur a alors plusieurs choix de réponse, correspondant à

plusieurs boutons cliquables. Les réponses sont souvent « oui » et « non », ainsi qu'un troisième choix, souvent « annuler », permettant de revenir à l'état avant la dernière action.

1.1.2.8.4. Fenêtre flottante (Utility window)

La fenêtre flottante, ou Utility window en anglais, est également appelée palette flottante ou barre d'outils flottante. C'est souvent une petite fenêtre, qui se situe par-dessus l'ensemble des autres fenêtres, de manière à être toujours visible. Celle-ci offre des outils ou des informations pour l'application active. Dans l'application de retouche d'image paint par exemple, la fenêtre flottante correspond à la fenêtre proposant des outils de modification de la police du texte.

1.1.2.9. Notion de Hiérarchie / arborescence

Les composants d'interface graphique sont soumis à une certaine hiérarchie, et ce à plusieurs niveaux :

- Au niveau de leur positionnement, au sein d'une fenêtre. En effet, les composants d'interface graphique sont aujourd'hui placés au sein d'une fenêtre de manière relative. Ainsi, un composant peut être positionné relativement à un autre composant, comme par exemple un une combo Box sera placée relativement à gauche par rapport à un label. De cette manière, l'ensemble des composants de l'interface n'ont pas de position absolue sur la fenêtre, mais sont placés relativement, ce qui va permettre une certaine forme de responsive sur la fenêtre.
  Complétée par l'intégration de divers composants graphiques au sein d'un composant de type conteneur, l'interface sera entièrement responsive. En effet, lorsque l'on redimensionnera la fenêtre en question, les champs se repositionneront en conséquence, proposant une fenêtre toujours lisible et parfaitement compréhensible.
- D'autre part, au niveau de leur conception. En effet, certains composants plus complexes sont des composés de composants plus simples. C'est par exemple le cas

de la zone de mot de passe, qui est en réalité une zone de texte avec des spécificités. Cela va se traduire au niveau du code par une classe zone de mot de passe héritant de la classe mère zone de texte.

1.2. Spécificités de chaque environnement

Bien que ces environnements possèdent certains fonctionnement semblables voire identiques sur certains points, X Window comme Win32 ont chacune des fonctionnements spécifiques que l'on explicitera.

1.2.1. Les spécificités de X Window

X11 possède quelques spécificités, telle que l'environnement de bureau souple, ou bien le fonctionnement client – serveur, ou encore ses différentes couches de programmation.

1.2.1.1. L'environnement de bureau souple

X window fournit à tous les systèmes d'exploitation fonctionnant avec Linux / UNIX un environnement de bureau bien plus souple que les autres systèmes d'exploitation ne possèdent. En effet, celui-ci se découpe en plusieurs éléments, au nombre de quatre. Il est constitué d'un système de fenêtrage, d'un gestionnaire de fenêtres, d'un environnement graphique et d'un gestionnaire de fichiers. Chacun d'eux peuvent être modifiables et interchangeables. Les principaux environnements de bureau, tels que Gnome, ou KDE, sont fournis avec une configuration de base, et permettent donc à l'utilisateur d'utiliser leur distribution Linux fraichement installée directement, sans avoir à effectuer de nombreux réglages. D'autres environnements de bureau plus spécifiques demandent un paramétrage plus ou moins complexe, qui nécessite tout de même de connaitre un minimum la structure de Linux.

Les programmes qui constituent les environnements de bureau ne sont pas tous directement visible par l'utilisateur. En effet, certains d'entre eux peuvent être des services

de bas niveau. L'architecture KDE par exemple propose une fonctionnalité qui donne à l'utilisateur un accès à un grand nombre de périphériques virtuels. Ces périphériques virtuels ne sont pas accessibles dans un environnement de bureau autre que KDE.

Voyons maintenant le rôle de chacun des sous-éléments de l'environnement de bureau de X Window.

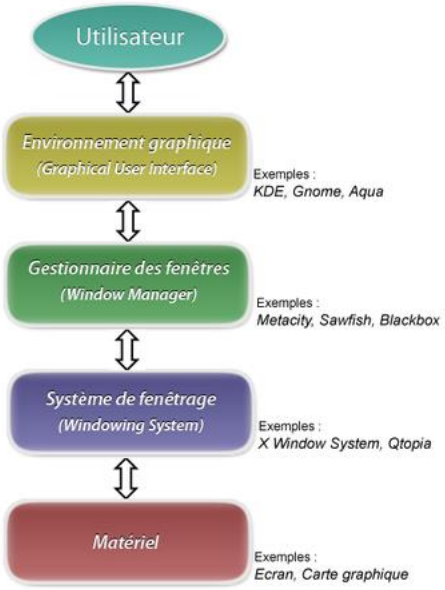

Schéma représentant les différentes couches sous X Window

1.2.1.1.1. Système de fenêtrage

Le système de fenêtrage, premier sous élément de l'environnement de bureau sous X11, est un logiciel allant permettre à l'utilisateur d'un ordinateur d'interagir avec plusieurs applications graphiques visibles simultanément, au travers des périphériques standard que sont le clavier et la souris. Chaque application va alors s'afficher dans une ou plusieurs fenêtres correspondant à des zones de l'écran de forme rectangulaires. L'utilisateur peut également déplacer et redimensionner les fenêtres, ou bien les fermer temporairement, ou encore les mettre en plein écran. Ces fenêtres peuvent se recouvrir les unes par-dessus les autres.

Au niveau technique, le système de fenêtrage est celui qui va fournir des éléments graphiques tels que le rendu des polices de caractères, le tracé de lignes. Il est à noter que ces primitives sont d'un niveau plus haut que les éléments abstraits fournis par le matériel graphique.

Le système de fenêtrage X permet à l'utilisateur de faire tourner l'application sur un ordinateur distant. Comme nous allons le voir dans la partie 1.2.1.2., c'est grace au X protocole que ceci est possible. Cette application distante se retrouve être cliente du serveur de fenêtrage qui est une application locale.

On peut citer quelques systèmes de fenêtrage :

- Qt extended, qui se base sur le langage Qt
- Quartz Compositor, qui est adressé à Mac OS X

1.2.1.1.2. Gestionnaire de fenêtre

Le gestionnaire de fenêtres, ou window manager en anglais, est un logiciel se chargeant de l'affichage et du placement des fenêtres d'applications.

Le gestionnaire de fenêtres va faire office d'intermédiaire entre le système de fenêtrage et l'environnement graphique.

Le gestionnaire de fenêtres étant lui-même un client sur serveur X, ce dernier offre des moyens de déplacement et de redimensionnement des fenêtres affichées par les autres clients. Il permet d'ajouter en outre une décoration aux fenêtres, en y ajoutant un cadre et une barre de titre. Il peut également modifier un fond d'écran. Ce gestionnaire de fenêtre peut proposer une gestion de la session utilisateur, ce qui va permettre de créer un historique et de garder une trace des actions effectuées par ce dernier.

Il est à préciser que la plupart des gestionnaires savent, de plus de ça, gérer les raccourcis clavier.

En outre, le gestionnaire de fenêtres permet d'ajouter les éléments graphiques suivants :

- Des icônes
- Des menus
- Des barres de tâches
- Des bureaux virtuels (avec la possibilité de passer d'un bureau à un autre via un raccourci clavier)

Attention, tous les gestionnaires de fenêtre ne gèrent pas toutes les fonctionnalités citées ci-dessus, ni l'ensemble des éléments listés.

1.2.1.1.3. Eléments d'environnement graphique

Les éléments de l'environnement graphique vont comprendre une combinaison de divers programmes et de librairies qui vont permettre la gestion du bureau.

1.2.1.1.4. Gestionnaire de fichiers

Le gestionnaire de fichiers est un logiciel qui propose une interface graphique qui va permettre d'interagir avec des fichiers. Plus précisément, il pourra effectuer un ensemble d'actions que sont la création, la lecture, la visualisation, l'impression, le renommage, le déplacement, la copie, la suppression, la recherche et la visualisation des propriétés de fichiers.

Les fichiers sont affichés le plus souvent sous un vue hiérarchique, avec des fichiers sous forme d'arborescence.

Les gestionnaires de fichiers peuvent contenir des fonctionnalités héritées des navigateurs web, telles que la navigation par les flèches du clavier. Une partie de ces gestionnaires de fichiers offrent le support de certains protocoles réseaux tels que FTP, Samba, ou encore NFS, qui va permettre de partager des fichiers entre ordinateurs.

1.2.1.2. Fonctionnement client – serveur

La grosse spécificité de X Window system vient de l'intégration du protocole X permettant le dialogue client – serveur entre ses différents sous- éléments.

Ce protocole est le résultat du fonctionnement des machines de l'époque. En effet, dans ses débuts, la structure d'un réseau de machines était différents d'aujourd'hui, où chaque PC possède sa puissance qui lui est propre. A l'époque, un réseau contenait un seul serveur puissant, relié à des terminaux sans puissance locale. L'ensemble des données et des applications étaient donc stockées et exécutés sur le serveur centralisé, qui s'occupait d'exécuter les programmes. L'opérateur avait un poste doté uniquement d'un clavier et d'un écran, avec éventuellement une souris. Le poste terminal contenait uniquement un boitier, auquel se connectaient les périphériques cités auparavant, et qui exécutait un serveur X. Le tout constituait alors un terminal X. Comme dit plus haut, les programmes s'exécutaient sur le serveur d'application centralisé, et l'IHM de ces programmes était gérée par le terminal X, c'est-à-dire le terminal client. Ce dernier peut être qualifié de client léger.

X fonctionne via le protocole IP. Il se base sur IPv4 mais il peut également fonctionner en IPv6.

Voici un exemple de dialogue client-serveur avec le protocole X :

- le logiciel serveur X est exécuté sur une machine de type client léger, c'est à dire terminal X ici. Ce terminal n'effectue que les tâches suivantes : réception et transmission des requêtes d'affichage, des saisies de texte et des déplacements de souris.
- Le logiciel client X est quant à lui un logiciel graphique qui va se connecter au logiciel serveur X. Il va alors lui envoyer ses requêtes d'affichages en utilisant le protocole X, tout cela au travers de la bibliothèque X. Le client regroupe uniquement l'application logicielle, qui utilise alors le protocole X pour déléguer au serveur X les taches d'IHM.

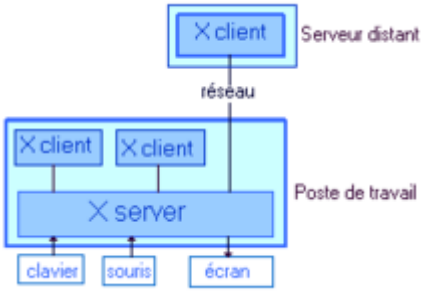

Schéma d'illustration d'un dialogue client-serveur sur X Window

Ce modèle de communication client – serveur va permettre également d'afficher les fenêtres et autres éléments d'interfaces graphiques en local et pas seulement via le réseau. Pour cela, les logiciels client et serveur sont installés sur la même machine.

Parmi tous les clients X, le gestionnaire de fenêtres est l'élément qui ressort le plus. En effet, son rôle comprend la gestion de l'affichage, ainsi que le déplacement, le redimensionnement et les décorations des fenêtres.

1.2.1.3. Les différentes couches de X Window

X se divise en plusieurs couches plus ou moins haut niveau. Nous les listerons de la plus basse couche à la plus haute :

- La plus basse couche est celle du protocole X. C'est à ce niveau-là que le client et le serveur s'échangent les données.
- Un peu plus haut, on trouve la couche de la bibliothèque X, dite également XLib. Cette librairie réalise la transition entre les données des requêtes, et les appels aux fonctions qui y correspondent. Elle peut également récupérer les évènements ayant lieu sur l'interface graphique, que ce soit une saisie clavier, un clic souris, ou même une modification de fenêtre. Il lui est également possible de créer, et de modifier des fenêtres, bien qu'à son niveau, ce ne soient que des rectangles où il est possible de dessiner dedans.
- Le niveau supérieur correspond au niveau de X Toolkit, aussi appelé Xt. C'est donc la boite à outil de composants graphiques par défaut. Cette boite à outils, qui propose

somme toute des widgets relativement basiques aujourd'hui, est de moins en moins utilisée. De nouvelles surcouches de XLib, plus récentes, sont utilisées à leur place. C'est le cas de Qt, ou bien GTK+ (The gimp toolkit).

1.2.2. Les spécificités de Win32

L'environnement Win32 de possède également certaines spécificités vis-à-vis du fonctionnement standard vu plus haut.

1.2.2.1. Fonctionnement de windows API

L'API Win32 de Microsoft est disponible et utilisée depuis le lancement de son système d'exploitation Windows 9X.

Cette API est en réalité un ensemble normalisé de fonctions qui vont permettre aux différents logiciels applicatifs d'utiliser les fonctionnalités du système d'exploitation Windows.

Cet ensemble de fonctions offre aux programmeurs la possibilité d'interagir avec le système d'exploitation de Microsoft. Ses possibilités sont immenses, et dépassent d'ailleurs de très loin celles apportées par les environnements de développement, tels que Windev ou Visual Basic. Ces API pourront par exemple permettre au développeur de contrôler une application, ou bien d'accéder au registre Windows, d'ouvrir d'autres applications, etc.

En réalité, ces API sont des recueils de fonctions très semblables à celles que tout le monde a l'habitude de rencontrer. Elles contiennent généralement un certain nombre de paramètres, et renvoient un certain résultat, ou bien effectuent une action précise. Ces fonctions sont contenues dans des fichiers dll, tels "gdi32.dll", "kernel32.dll", etc. Les fonctions les plus couramment utilisées sont celles qui constituent Microsoft Windows lui-même.

L'ensemble des fonctionnalités de Windows API peuvent être classifiées dans les catégories suivantes :

- Les services de base
- Les boites de dialogues communes
- La bibliothèque de contrôles communs
- Le shell Windows
- L'interface graphique
- L'interface Utilisateur
- Les services réseaux
- Les autres API liées

Intéressons-nous maintenant aux catégories ainsi qu'aux fichiers dll qui contiennent les fonctions permettant la gestion de l'interface graphique sur un système d'exploitation Windows.

Nous nous intéresserons ici aux catégories de l'interface graphique, de l'interface utilisateur, et plus particulièrement aux fichiers gdi32.dll et user32.dll, qui regroupent l'ensemble des fonctionnalités nécessaire au fonctionnement de l'interface graphique et de son interaction avec l'utilisateur.

Gdi32.dll va permettre l'accès aux ressources pour l'affichage sur les périphériques de sortie comme les écrans, et les imprimantes.

GDI signifie Graphics Device Interface ou Graphical Device Interface en anglais, et qui se traduit en français par Interface des dispositifs d'affichage. Il fait partie des trois composants fondamentaux du système d'exploitation Microsoft Windows, avec le kernel et l'user (contenus dans les fichiers kernel32.dll et user32.dll).

GDI est une norme de Microsoft Windows pour la représentation d'éléments graphiques d'une part, mais également pour leur transmission aux périphériques de sortie d'autre part.

GDI va donc proposer un certain nombre de fonctionnalités qui vont principalement afficher des lignes et des courbes, ou bien récupérer le rendu des polices d'écriture. Ce n'est par contre pas lui qui se chargera de l'affichage des fenêtres, des menus et autres contenus graphiques. Cette tâche est exécutée par le User, qui est contenu dans le fichier user32.dll.

Parmi l'ensemble de ses méthodes ayant un accès direct au matériel, les fonctionnalités les plus impressionnantes de GDI sont probablement sa capacité d'abstraction du matériel de sortie, ainsi que ses capacités vectorielles. En effet, l'utilisation de GDI va permettre de dessiner sur de multiples périphériques, sans se préoccuper particulièrement de ceux-ci. De ce fait, les dessins réalisés seront reproduits très fidèlement, que le périphérique de sortie soit un écran ou une imprimante. Cette capacité est commune à toutes les applications WYSIWYG de Microsoft Windows. Ce sigle WYSIWYG signifie What you see is what you get en anglais, et littéralement ce que vous voyez est ce que vous obtenez en français.

Cependant, l'API de GDI va être rapidement obsolète lorsque l'interface graphique demandera des rendus graphiques élevés. Bien que certains jeux relativement simples et peu gourmands en puissance graphique tels que le démineur utilisent GDI, les autres jeux ou autres logiciels très consommateurs utiliseront une autre API. En outre, GDI ne supporte pas les fonctionnalités en 3D. Pour ces besoins précis, ce sont les API DirectX et OpenGL qui vont y répondre. Celles-ci sont en effet conçues pour exposer les fonctions matérielles 3D aux développeurs.

Il est à noter que GDI a à plusieurs reprises pu être accéléré matériellement. En effet, entre Windows 95 et Windows XP, GDI a eu un rôle d'intermédiaire entre l'application et le driver graphique. GDI redevient entièrement logiciel sous Windows Vista, alors que depuis l'arrivée de Windows 7, les applications sont à nouveau accélérées matériellement, d'une autre manière plus performante.

GDI permet en outre d'imprimer au travers d'un type d'imprimante particulier. Ces imprimantes compatibles, également appelées GDI, sont des périphériques externes qui physiquement ne possèdent ni RAM, ni CPU, et très peu d'électronique, en comparaison à une imprimante plutôt standard.

En effet, l'imprimante et l'ordinateur utilisent tous deux GDI afin de communiquer. L'image d'origine est transformée en format bitmap, puis simplement envoyée telle quelle à l'imprimante, qui édite le document.

2. Objectifs d'une interface

Une interface graphique se doit de répondre à un certain nombre de critères afin

2.1. Pattern d'organisation

Afin d'offrir une performance maximale, une application doit mettre en place des structures allant permettre de gérer de manière optimale les interactions entre l'interface graphique et le traitement des informations.

Les design patterns (ou modèles de conception) ont donc émergés dans cette optique.

Le patron de conception peut être considéré comme un arrangement caractéristique de modules qui a été reconnu au préalable comme bonne pratique, et ce en réponse à un problème de conception précis. Les patrons de conception sont donc considérés comme des solutions à des problèmes de conception précis réutilisables à l'infini.

Un patron de conception est issu de l'expérience des concepteurs de logiciels.

2.1.1. Pattern Modèle-Vue-Contrôleur (MVC)

Le design pattern Modèle-Vue-Contrôleur (MVC) est un pattern architectural qui sépare les données (le modèle), l'interface homme-machine (la vue) et la logique de contrôle (le contrôleur).

Ce modèle de conception impose donc une séparation en 3 couches distinctes, telle que l'illustre le schéma ci-dessous :

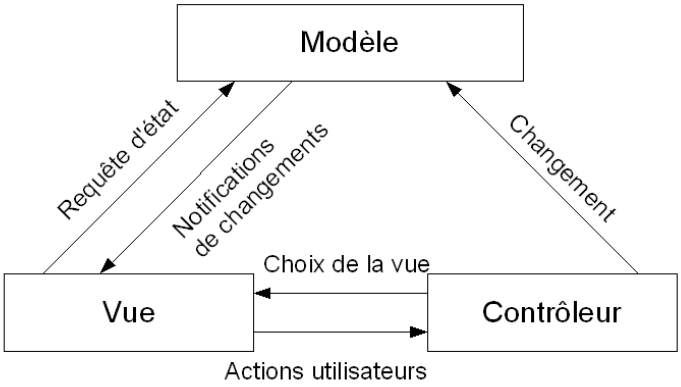

Schéma du design pattern Modèle-Vue-Contrôleur

Le modèle : Il représente le noyau algorithmique de l'application, à savoir l'ensemble des traitements de données de l'application, ainsi que les interactions avec la base de données. Celui-ci rassemble l'ensemble du métier de l'application, c'est-à-dire la gestion des données, et de leur intégrité. Ce modèle comporte un ensemble de traitements permettant l'ajout, la modification, la suppression et la récupération de données, dans l'application comme dans la base de données. Le modèle ne permet d'avoir aucun lien direct vers le contrôleur ou la vue. Sa communication avec la vue s'effectue au travers d'un mécanisme d'événements et de déclencheurs, mis en place à l'aide du patron de conception Observateur.

La vue : Elle représente l'interface utilisateur, ce avec quoi celui-ci interagit. Elle possède 2 objectifs distincts ; Son premier objectif est de présenter les résultats renvoyés par le modèle, alors que son second est de recevoir toute action de l'utilisateur (survol d'une zone / d'un élément, clic de souris, sélection d'un bouton radio, cochage d'une case, entrée de texte, etc.). Ces différentes actions enclenchent des événements qui sont envoyés au contrôleur. Il est à noter que la vue n'effectue pas de traitement, elle se contente simplement d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur. Rien n'empêche plusieurs vues de présenter les données d'un même modèle.

Le contrôleur : Il gère l'interface entre le modèle et la vue. Il a pour rôle la gestion des événements, afin de mettre à jour la vue ou le modèle, et de les synchroniser. Il effectue la synchronisation entre le modèle et les vues. C'est lui qui reçoit l'ensemble des événements

effectués par l'utilisateur sur la vue, et qui enclenche les actions à réaliser. Dans le cas d'une action nécessitant une mise à jour de données, le contrôleur demande au modèle la modification des données, et notifie également la vue que les données ont été modifiées, pour que celle-ci se mette à jour et actualise les données modifiées. Le contrôleur ne modifie aucune donnée et n'effectue aucun traitement lui-même. Il traite simplement les évènements de la vue et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande.

On peut résumer le fonctionnement du pattern MVC par l'enchainement suivant :

- Un évènement déclenché par une action de l'utilisateur sur la vue est récupéré puis analysée par le contrôleur.
- Le contrôleur demande alors au modèle lui étant associé d'effectuer des traitements correspondants, et notifie à la vue que la requête a bien été traitée.
- la vue, une fois notifiée, effectue alors une requête directement au modèle afin d'actualiser ses données.

Il est à noter que la synchronisation entre la vue et le modèle s'effectue au travers d'un système d'évènements mis en place par le pattern Observer, ou Observateur en français. Ce dernier permet, lors d'une modification du modèle, de générer des événements et d'indiquer à la vue qu'il faut se mettre à jour.

Ce modèle de conception apporte 2 avantages majeurs :

- La séparation des différentes couches. Une couche peut tout à fait être modifiée sans altérer les autres. C'est-à-dire que comme toutes les couches sont clairement dissociées, on pourra en modifier une pour, par exemple, remplacer le Framework graphique de la couche vue sans porter atteinte aux autres couches. On pourrait également changer le modèle sans avoir à toucher à la vue et au contrôleur. Cela rend les modifications plus simples.
- La synchronisation des différentes vues affichées. Avec ce design pattern, toutes les vues qui font appel aux mêmes données sont synchronisées automatiquement.

Il faut tout de même garder en mémoire, que la mise en œuvre de MVC dans une application peut s'avérer relativement difficile. En effet, ce modèle de conception introduit tout de même un niveau de complexité assez élevé. De plus, implémenter MVC dans votre application nécessite une bonne conception initiale de l'application et de sa structure. Ce pattern plutôt adressé aux moyennes et grandes applications.

Aujourd'hui, de nombreux langages / Framework utilisent MVC, listons les plus connus :

- En JAVA : Struts, Spring MVC, Swing, SWT, PureMVC, etc
- En C++ : Qt
- En ASP.NET : ASP.NET MVC
- En Objective C : Cocoa
- En PHP : CakePHP, Symfony, Zend Framework, Joolma, etc
- En Ruby : Ruby on Rails

2.1.2. Pattern Modèle-Vue-Présentateur (MVP)

Le patron de conception Modèle-Vue-Présentateur est un modèle dérivé du patron Modèle-Vue-Contrôleur (MVC). Il reprend la structure du modèle MVC à quelques différences près. En effet, le présentateur prend la place du contrôleur, et obtient le rôle de « middle man », c'est-à-dire qu'il va jouer seul le rôle d'intermédiaire entre le modèle et la vue.

Le modèle Modèle-Vue-Présentateur définit trois différents types de rôles, comme l'illustre le schéma ci-dessous :

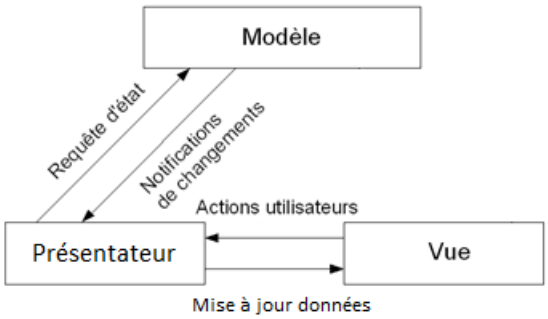

Schéma d'illustration du patron de conception Modèle-Vue-Présentateur

Le modèle : De la même manière que le modèle du patron de conception MVC, le modèle représente ici le noyau algorithmique de l'application, à savoir l'ensemble des traitements de données de l'application, ainsi que les interactions avec la base de données. Celui-ci rassemble l'ensemble du métier de l'application, c'est-à-dire la gestion des données, et de leur intégrité. Ce modèle comporte un ensemble de traitements permettant l'ajout, la modification, la suppression et la récupération de données, dans l'application comme dans la base de données.

La vue : Comme pour la vue du modèle MVC, la vue représente ici l'interface utilisateur, ce avec quoi celui-ci interagit. Elle possède 2 objectifs distincts ; Son premier objectif est de présenter les résultats renvoyés par le présentateur, alors que son second est de recevoir toute action de l'utilisateur.

Le présentateur : C'est la partie intermédiaire qui communique avec les deux autres éléments que sont la vue et le modèle. Ce présentateur permet d'une part de traduire et transmettre les commandes de l'utilisateur envoyées de la vue vers le modèle, et d'autre part de formater et afficher les données du modèle dans la vue.

Le principe est de découpler la vue et le modèle, en utilisant le présentateur comme intermédiaire.

Plusieurs vues d'un même modèle peuvent être présentées simultanément. Quand l'utilisateur interagit avec une vue pour effectuer une modification, la vue transmet la requête au présentateur. Celui-ci la transmet au modèle, en effectuant généralement une transformation des paramètres, ou en appelant une ou plusieurs méthodes du modèle. Puis le présentateur notifie toutes les vues afin de les mettre à jour pour prendre en compte la modification effectuée.

L'avantage ici du MVP vis-à-vis du modèle MVC, est une mise en place de tests unitaires plus aisée, ainsi qu'une meilleure séparation des différents éléments dans la logique de présentation.

2.1.3. Pattern Modèle-Vue-VueModèle (MVVM)

Le patron de conception Modèle-Vue-VueModèle est un patron similaire aux patrons vus précédemment qui a été développé par Microsoft. Son but est de proposer une méthode de conception adapté à ses propres outils de développement. En effet, MVVM est adapté aux applications développées en utilisant le XAML, mais également les applications développées avec les technologiques Windows Presentation Foundation (WPF), Silverlight, Windows Phone et Windows 8 notamment. Encore une fois, cette structure a pour objectif de séparer la vue de la logique, ainsi que de l'accès aux données, tout en utilisant un système de déclenchement d'événements. Elle permet également d'architecturer efficacement une application, ainsi que d'en faciliter sa testabilité, et son évolutivité.

Le modèle Modèle-Vue-VueModèle définit lui aussi trois différents types de rôles, comme l'illustre le schéma ci-dessous :

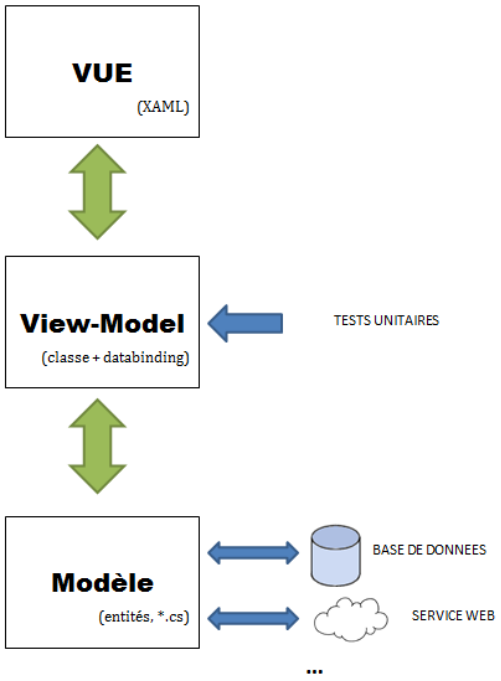

Schéma d'illustration du patron de conception Modèle-Vue-VueModèle

Le modèle : Le modèle correspond au noyau algorithmique de l'application, à savoir l'ensemble des traitements de données de l'application, ainsi que les interactions avec la base de données. Celui-ci rassemble l'ensemble du métier de l'application, c'est-à-dire la

gestion des données, et de leur intégrité. Ce modèle comporte un ensemble de traitements permettant l'ajout, la modification, la suppression et la récupération de données, dans l'application comme dans la base de données

La vue : La vue représente l'interface utilisateur. Elle possède divers objectifs ; Son premier objectif est de présenter les résultats renvoyés par la VueModèle , alors que son second est de recevoir toute action de l'utilisateur. Ces différentes actions enclenchent des événements qui sont envoyés au VueModèle . Il est à noter que la vue n'effectue pas de traitement, elle se contente simplement d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur. En pratique, la vue correspond ici au fichier .xaml.

Le xaml est basé sur le langage XML. Celui-ci permet au développeur de développer une application client Windows aussi aisément qu'une application web. Le concept est de séparer la déclaration des objets d'un programme du code sous-jacent comme cela est déjà le cas dans les applications Web de type ASP.NET.

Lorsque l'on utilise le xaml pour WPF, le développement de l'interface utilisateur peut être confié à des designers, et donc des métiers non techniques.

La VueModèle : Pouvant être traduit par « modèle de vue », cet élément correspond à la liaison entre la vue et le modèle. Il s'agit d'une classe qui fournit une abstraction de la vue. Ce modèle de vue, s'appuie sur la puissance du Data binding pour mettre à disposition de la vue les données du modèle. Le binding, ou liaison en français, est un mécanisme allant permettre de générer un objet à partir de la lecture d'un document de type XML. Le Data binding, lui, permet de lier des objets entre eux pour les faire communiquer. Ici, la VueModèle utilise le Data binding pour lier la vue et le modèle.

Le MVVM reprend la plupart des caractéristiques des patterns présentés préalablement. Il propose en effet les avantages d'une dissociation des différentes couches, permettant une meilleure maintenabilité de l'application. De la même manière que le pattern MVP, ce patron de conception va de plus permettre une réelle facilité de mise en place de tests unitaires.

On peut avancer le fait que Microsoft s'est fortement inspiré du patron de conception MVP pour réaliser le sien fait « maison » comme cette société à l'habitude. De cette manière, le MVVM est adapté et destiné à ses outils propres.

2.2. Les différents critères à prendre en considération

En plus d'être pensée lors du choix d'utilisation de l'architecture globale d'une application, un projet doit répondre à plusieurs critères afin que son interface graphique y réponde aussi.

En effet, une interface graphique doit répondre aux critères suivants :

- Son accessibilité par la plus grande partie de son public, que ce soit au niveau du support sur lequel on consulte l'application, ou au niveau des différences entre les profils de personnes qui visitent, ou utilisent l'application, web ou non. Ces personnes peuvent être malvoyantes, atteint d'une surdité, etc. Tant d'éléments qu'il faut prendre en compte lors de la réalisation d'une application.
- Sa propriété responsive afin de s'adapter et d'être visible, disponible et compréhensible à l'ensemble des terminaux et des résolutions d'écrans existants.
- Sa testabilité automatisée afin d'obtenir constamment une application performante et sans bugs ni anomalies.
- Son critère modifiable par l'ensemble des corps de métiers ayant à travailler au sein du projet, qu'ils soient techniques ou non.

2.2.1. Accessibilité

L'accessibilité numérique va concerner deux catégories d'applications. D'un côté il y a les applications dites natives, ou client lourd. Ce sont les applications installées sur nos appareils numériques. La deuxième catégorie d'application va concerner ce qu'on appelle les applications web. Celles-ci sont accessibles via un navigateur, tels que Google Chrome,

Internet Explorer ou bien Mozilla Firefox. Nous nous intéresserons dans un premier temps à ce second type d'application, à savoir les applications Web.

Il ne faut jamais perdre de vue pour qui il faut rendre les applications, qu'elles soient web ou non, accessibles :

Pour un aveugle, qui ne peut avoir de vision d'ensemble de la page, celui-ci nécessite des informations structurées. Il peut se retrouver bloqué, sans savoir où il se situé au niveau de la navigation, car sa plage braille ne lui renvoie plus aucune information.

Pour une malvoyante qui utilise une synthèse vocale ou un logiciel de revue d'écran, il est frustrant d'entendre que la description d'une image est un nom générique sans signification avec ce que l'image elle-même représente.

L'accessibilité peut également se transcrire par une insertion de sous-titres dans une vidéo, et ce afin que les malentendants comprennent entièrement le contenu de celle-ci.

Une application accessible doit en outre prendre en considération les handicapés moteurs, qui utilisent parfois des périphériques de navigation différents du clavier et de la souris, par la mise en place d'une interface graphique structurée. Cette GUI structurée se traduit par un balisage clair et normalisé.

Ces exemples ne traduisent bien sûr pas l'ensemble du public, mais plutôt la diversité de celui-ci. Cela permet de prendre en considération qu'il y a un grand nombre de manières différentes de naviguer dans une application, et que l'application développée s'adresse cet ensemble.

2.2.1.1. Mise en place de l'accessibilité dans un projet

L'accessibilité doit être pensée et prise en compte dès le début d'un projet. Lors de l'élaboration d'un projet, les spécifications techniques doivent être précisées dans le but de rendre le projet accessible.

Il est également nécessaire que cette dimension d'accessibilité soit prise en considération durant chaque étape / modification du projet, que cela soit lors de la création, lors d'une évolution ou bien d'une restructuration complète de l'ensemble du projet.

C'est là tout l'enjeu d'une application accessible : faire entrer la notion d'accessibilité à toutes les étapes et dans tous les esprits. Il donc faut que quelqu'un s'en charge très tôt dans un projet.

2.2.1.2. Reconnaitre une application accessible

Le Référentiel Général de l'Accessibilité pour les Administrations (RGAA) est un référentiel concernant l'accessibilité et qui s'applique aux sites de communication publics. Ce dernier reprend l'organisation du WCAG 2.0, ce qui permet de faire ressortir 4 grands principes.

Etudions ces grands principes :

Premier principe : il faut que les contenus soient perceptibles. C'est à dire que « L'information et les composants de l'interface utilisateur doivent être présentés à l'utilisateur de façon à ce qu'il puisse les percevoir. ». Cela signifie la mise en place d'alternatives textuelles aux images, ainsi qu'à tous les éléments visuels, comme les boutons graphiques, ou bien la description textuelle d'un graphique. De la même manière, on peut penser à la mise en place captcha audio, ainsi qu'à la mise en place d'alternatives au sous-titrage et à l'audiodescription pour les contenus audio ou vidéo. Il s'agit également de séparer le contenu de la présentation via l'utilisation de feuilles de style.

Deuxième principe : Les « composants de l'interface utilisateur et de navigation » doivent être utilisables. Cela signifie qu'il faut que toutes les fonctionnalités soient accessibles au clavier. Des éléments d'orientation doivent également être présents. Un lecteur qui utilise une synthèse vocale lit de manière séquentielle, chronologique. Il ne lui est donc pas possible d'avoir une appréhension globale des informations. Des éléments d'orientation doivent donc lui permettre de repérer la hiérarchie des informations, pour aller directement à l'information qui l'intéresse et éviter s'il le souhaite certaines informations. Il est

également important de laisser suffisamment de temps à l'utilisateur pour lire le contenu et pour mener les actions nécessaires. Il est également conseiller d'éviter les flashs susceptibles de déclencher des crises d'épilepsie chez certains utilisateurs.

Troisième principe : Les informations et l'utilisation de l'interface utilisateur doivent être compréhensibles. Ainsi, par exemple, il faut qu'un dispositif technique de lecture sache si le texte est dans une langue étrangère au risque de produire un texte incompréhensible. Il faut également faire en sorte que la structuration se répète afin de rendre la navigation prévisible.

Quatrième principe : Le contenu doit être suffisamment fiable pour être interprété de manière fiable par un large panel d'utilisateurs différents, y compris les technologies d'assistance. Il faut donc veiller à ce que les technologies d'assistance utilisent correctement les contenus balisés.

2.2.1.3. La certification d'un site accessible

Les recommandations WCAG 2.0 se répartissent selon 3 niveaux :

Le niveau A : niveau fondamental satisfaisant tous les critères d'accessibilité de priorité 1. Pour une conformité de niveau A (le niveau minimal), la page Web satisfait à tous les critères de succès de niveau A ou une version de remplacement est fournie. Par exemple, un texte explique une vidéo ou une image.

Le niveau AA : niveau satisfaisant tous les critères d'accessibilité de priorité 1 et 2. Le site offre un accès « correct » aux informations contenues dans les documents Web. Par exemple, la possibilité de recourir à l'agrandissement des textes sans perte d'information et sans avoir recours à une technologie d'assistance.

Le niveau AAA : niveau satisfaisant tous les critères d'accessibilité de priorité 1, 2 et 3. Le site offre un accès excellent aux informations contenues dans les documents Web. Par exemple, une description audio étendue met en pause la vidéo pour expliquer l'ensemble des informations nécessaires à la compréhension de l'élément.

Il est à noter que le niveau recommandé au niveau européen est le niveau AA. C'est également le niveau de conformité que doivent atteindre les sites publics français.

2.2.2. Responsive

Lorsque l'on emploie le terme responsive, on veut aujourd'hui souvent dire Responsive Web design (ou RWD en abrégé). En effet, la dynamique actuelle du responsive s'adresse principalement aux sites internet et applications web disponibles par navigateur. Nous allons donc bien sur parler ici principalement des applications web, mais il ne faut pas oublier les applications dites client lourd, par contradiction aux applications web. En effet, ces dernières ont également une certaine nécessité d'être responsive. Cependant, ces applications lourdes sont dans leur plus grande majorité, naturellement responsives, car elles utilisent les derniers langages et / ou librairies qui utilisent la notion de conteneurs, ce qui permet de placer dans l'interface de l'application des composants de manière tout à fait relative vis-à-vis des autres, et non plus de manière statique comme auparavant, où l'on précisait de manière statique la position ainsi que la dimension de chaque composant. Aujourd'hui, une application sera toujours lisible et compréhensible lorsque l'on redimensionne la fenêtre de l'application lourde.

Illustration d'une interface responsive

2.2.2.1. Définition

De manière globale, le responsive signifie, dans le cadre d'une application, de posséder une interface qui sera toujours lisible et compréhensive, et ce même lorsque les dimensions de l'application sont modifiées.

Les applications accessibles via navigateur étant aujourd'hui les plus utilisées et les plus demandées, il est intéressant de mentionner la notion de Responsive Web Design, ou RWD en abrégé. En effet, le Responsive Web Design, ou conception de sites web adaptatifs en français, est une approche de conception de sites ou application Web dans laquelle ces derniers sont conçus pour offrir au visiteur une expérience de consultation optimale facilitant la lecture et la navigation. Le gros avantage ici est le fait que l'utilisateur peut consulter le même site Web à travers une large gamme d'appareils, que ce soient des moniteurs d'ordinateur, des smartphones, des tablettes, des TV, etc. A chaque fois, le site Web apparaitra avec le même confort visuel, sans avoir besoin de zoomer ni de se déplacer sur les appareils tactiles, manipulations qui dégradent considérablement l'expérience utilisateur. En outre, l'intérêt ici est que l'interface graphique du site / application Web en question n'est pas réalisé un nombre de fois égal au nombre de résolutions d'écrans différents, mais qu'une seule fois, et s'adapte automatiquement en fonction de cette résolution. Cette technique s'appuie sur la technologie CSS3 récemment sortie, et de l'outil phare de Twitter qu'est BootStrap.

2.2.2.2. Utilisation

Le “responsive web design” est né suite à un besoin grandissant. De nos jours, il n’y a pas un client qui demande un site Internet sans demander une version mobile de celui-ci. Seulement en quelques années le nombre d’appareils et de résolutions servant à consulter des sites web ont tout simplement explosé : ordinateurs, smartphones, tablettes, web TV, format portrait, paysage, etc. Entre 2005 et 2008 on a identifié plus de 400 résolutions d’écran différentes pour tous les appareils vendus. La technique de développer autant de versions d’un site qu’il y a de résolutions différentes n’est alors plus envisageable, tant le travail est fastidieux. La solution à ce problème grandissant s’est donc imposée d’elle-même : le responsive design.

Au niveau technique, le responsive fait appel aux styles CSS, ainsi qu’à des balises dites media queries, qui vont permettre à la page de charger différentes règles CSS en fonction de la résolution de l’écran. Le style de l’application web se met alors automatiquement à jour en fonction de la résolution de l’écran, ce qui va lui permettre d’être toujours lisible et compréhensible. Attention tout de même, il est nécessaire que l’interface de l’application en question ait été correctement développée. Le responsive n’est pas magique, c’est un outil puissant mais qui nécessite d’être maitrisé pour être utilisé de la bonne manière.

Rapidement, divers Framework vont voir le jour afin de faciliter cette mise en place du responsive. Le projet BootStrap de la firme Twitter va attirer l’œil de beaucoup et va devenir rapidement un des plus populaires du net. Il est aujourd’hui la solution incontournable afin de rendre une application web responsive.

2.2.3. Testable

Il est intéressant de noter qu’une interface graphique peut être testée par n’importe qui. Ce qui signifie qu’une interface graphique sera toujours testable. Ici, il faut entendre par testable, le test automatique, facilement exécutable, et de manière redondante, ce de manière automatisée. C’est à ça qu’il faut entendre quand on emploiera le terme testable.

Comme partout aujourd'hui, il est nécessaire qu'une application subisse un contrôle qualité. Les tests unitaires et fonctionnels existent pour le code métier de l'application, mais il est également nécessaire de mettre en place des tests sur l'interface graphique de l'application.

Que ce soit une demande client ou de la rigueur des développeurs, les tests d'interface sont là pour palier au risque d'erreur. En effet, comme tout ce qui est artificiel, les programmes informatiques sont également sujets à des erreurs. La complexité du code métier de l'application peut amener à des conflits inattendus dans le programme lui-même. Un seul défaut sur un élément graphique peut entrainer des conséquences néfastes. Il n'y a qu'à prendre exemple sur un bouton de validation disparaissant, bloquant tout un processus de validation de formulaire.

L'objectif de ces tests de l'interface graphique va être de vérifier les éléments de cohérence, la pertinence et l'efficacité des composants d'interface graphique.

- La cohérence des éléments d'interface graphique signifie que la même position géographique, ainsi que leur fonction s'effectue bien au sein de l'application.
- La pertinence et l'efficacité d'un élément de l'interface graphique signifient que celui-ci est utile à l'utilisateur. Au travers de la pertinence, on peut vérifier qu'il n'y ait pas d'éléments inutiles ou redondants dans l'interface graphique, tel qu'un surplus de fonctionnalités au même endroit, avec un certain nombre de boutons ou d'autres éléments jamais utilisés.

Plusieurs solutions logicielles ayant ce rôle de rédiger et exécuter ces tests existent. C'est le cas de Testkink par exemple. Dans celui-ci, on rédige les cas de tests, étape par étape, avec une manipulation à effectuer, et un état attendu. Généralement, ces tests sont rédigés à chaque nouvelle fonctionnalité, et exécutés à chaque livraison. Cependant ces tests nécessitent une personne pour les réaliser.

2.2.3.1. Les tests automatisés

Ces tests de l'interface graphique sont très fastidieux et chronophages. Il est donc intéressant d'envisager l'automatisation comme une option à prendre. L'automatisation des tests graphiques suggère qu'une autre application, séparée et distinct du programme à tester, va vérifier chaque aspect graphique du programme à tester. Celui-ci devra tester principalement les actions que les utilisateurs finaux réaliseront tous les jours. Il faut donc créer des cas de tests, accessoirement avec le client afin qu'il détermine comment il utilise l'application. Ces cas de tests peuvent alors être automatisés. Il doit en outre tester les actions standards suivantes : le démarrage du programme, les erreurs de manipulation qu'un utilisateur peut effectuer par inadvertance, et l'accès à bases de données, la génération de rapports, etc. Ce logiciel de test d'interface graphique automatisé va simuler la vie réelle d'utilisation du programme. Ce type de logiciel va tester tous les cas possibles. Le processus est réitéré à chaque modification du code par les développeurs pour vérifier la cohérence et l'intégrité du programme.

2.2.3.2. Les logiciels existants

Plusieurs logiciels de ce type existent. Certains sont payants, d'autres gratuits, et sont plus ou moins performants. De plus, ils ne peuvent parfois s'appliquer qu'à certains langages de programmation.

Sélénium est un bon exemple. Il se base sur les propriétés identifiants des champs dans le HTML pour réaliser les tests d'interface. Cela permet de ne pas avoir à se baser sur une seule résolution d'écran, et n'oblige pas à re-paramétrer les tests lorsqu'un élément de l'interface graphique est déplacé.

Sikuli, quant à lui, est un logiciel open source qui ressemble plus à un exécuteur de macro, c'est-à-dire une suite d'actions. Il se base uniquement sur l'interface graphique. De ce fait, il est nécessaire de toujours avoir la même résolution pour exécuter les tests. En outre, si l'on déplace un champ, il faudra re-paramétrer le test également. Coté avantages, il peut

s'adresser à un vaste panel de technologies de programmation et de langages, celui-ci se basant uniquement sur l'interface sans se préoccuper du code derrière.

2.2.4. Personnalisable

La personnalisation d'une interface graphique est aujourd'hui d'autant plus d'actualité que la plupart des projets développés aujourd'hui ne créent pas tout de zéro. En effet, tout projet est développé à partir de Framework et de librairies, pour des raisons de gain de temps et de réutilisabilité du code propre. Que ce soit un site internet développé à partir du CMS Joomla par exemple, ou une application web développée en Java / JEE avec un Framework, les interfaces graphiques auront une structure et une charte graphique de base. Il est donc évident que chacun souhaite personnaliser avec ses couleurs, et ses composants graphiques, l'interface de son projet. Il est possible de modifier directement dans le code de ces structures utilisées toutes faites certains paramètres afin de changer l'interface à sa convenance, cependant cela n'est pas très pratique. D'une part techniquement, car le code en question n'est généralement pas maitrisé par le développeur, étant donné que ce dernier n'a pas développé le Framework utilisé. D'autre part, car ces outils ont mis en place une possibilité de personnalisation de l'interface graphique adaptée.

En effet, un grand nombre de Framework et structures de bases pour sites internet (Joomla, wordpress, drupal, etc.) proposent la personnalisation de leur interface à travers des menus, ou des fichiers de configuration externes. De plus, les communautés regroupées autour de ces outils proposent un certain nombre de thèmes visuels différents qui vont permettre de personnaliser encore plus facilement le contenu graphique. En outre, un grand nombre de documentations et tutoriels proposent d'expliquer et de suivre pas à pas les démarches à réaliser, afin même qu'une personne non technique puisse personnaliser à sa guise les graphismes d'un site ou d'une application web.

On peut également parler de personnalisation de l'interface d'une application dite client lourd, en important ou créant de nouveaux graphismes destinés aux composants d'interface graphique. Il suffit de modifier le widget toolkit utilisé, qui gère les composants, et donc les formes et couleurs qui les composent.

2.2.5. *Modifiable / Evolutif*

On entend par interface graphique modifiable, une interface graphique pouvant être utilisée, réalisée, modifiée, maintenue, etc. par tous les corps de métiers qui auront nécessité d'y avoir accès.

L'application, et plus particulièrement son interface graphique, doit être évolutive. Cela signifie qu'elle peut être modifiée sans avoir un impact sur le reste de l'application. Par exemple, le designer va pouvoir modifier l'interface sans que le code métier de l'application ne soit impacté.

De la même manière, le développeur, en rajoutant des fonctionnalités dans le code métier de l'application, ne devra pas impacter obligatoirement l'interface graphique. Il est à noter que la mise en place d'une structure de type design pattern vue plus haut ne sera que bénéfique à la maintenance et l'évolutivité de l'application, ainsi que de son interface.

2.2.6. Les métiers concernés par l'interface graphique

Au sein d'un projet informatique, plusieurs corps de métiers cohabitent afin de réunir l'ensemble des compétences nécessaires à sa réalisation complète. En effet, le domaine informatique est tellement vaste qu'il est difficile pour une personne de maitriser l'ensemble des compétences demandées à un professionnel, que ce soit des compétences en développement informatique, des compétences en design et graphisme, ainsi que des compétences en intégration de contenu. De nombreux métiers ont été créés, et ce sont plus ou moins spécialisés dans certains domaines. De plus, en fonction de la taille de l'entreprise dans laquelle on travaille, et du domaine informatique dans lequel on évolue, les métiers impliqués seront différents, ainsi que les charges de travail affectés aux uns ou aux autres corps de métiers. En effet, dans les entreprises de communication proposant la création de sites internet, les types de métiers vont être relativement simples à délimiter, et souvent moins spécialisés que des profils recrutés dans les grandes SSII et associés à des projets d'envergure, où les contraintes techniques, fonctionnelles, contractuelles et légales ne sont pas les mêmes.

Nous allons simplifier ici les corps de métiers afin d'en faire ressortir les trois principaux qui vont être concernés par l'interface graphique d'un projet :

- le développeur : celui qui programme, qui va créer le métier de l'application.
- Le graphiste : celui qui va s'occuper de créer les images, l'ensemble des éléments graphiques nécessaires au projet.
- L'intégrateur : celui qui va intégrer dans le projet, les différents éléments graphiques.

2.2.6.1. Le développeur

Un développeur est un informaticien qui réalise des logiciels ou des applications destinés à paraître sur Internet ou sur client lourd, c'est-à-dire une application installable sur un système d'exploitation Windows, Linux/UNIX ou Mac, en créant des algorithmes et en les mettant en œuvre dans un langage de programmation prédéfini.

On peut aujourd'hui distinguer une nette tendance à développer des applications accessibles via navigateur, qu'elles soient destinées au grand public via internet ou à un intranet d'entreprise. Cette tendance s'explique par le fait qu'un navigateur est multiplateforme, ce qui va permettre à une application d'être accessible quel que soit le système d'exploitation de la machine utilisée pour y accéder, ce qui est un grand plus vis-à-vis d'une application dite client lourd, qui n'est développée et donc disponible et compatible uniquement sur un seul système d'exploitation.

La plupart des développeurs peuvent être aujourd'hui catégorisés en deux parties distinctes :

- Les développeurs Web, qui développent des sites internet, et qui utilisent les langages de type PHP, ou ASP, ainsi que les langages du web que sont HTML / CSS, le JavaScript, et tous les framework et autres outils. Ils vont être amenés à utiliser également des CMS (Content Management System ou système de gestion de contenu en français), qui sont des logiciels destinés à la conception et à la mise à jour dynamique de sites Web.

- Les développeurs travaillant dans l'informatique de gestion, qui développent des applications pour des clients professionnels, et qui aujourd'hui ont souvent comme contrainte technique d'être accessible via navigateur par intranet ou internet. Ces développeurs utilisent principalement le Java / JEE comme langage et architecture de conception, qui va comprendre une partie dite métier dans laquelle sera contenu l'ensemble des algorithmes et des traitements de l'application. La base de données sera également sur serveur distant. Bien que la technologie soit différente de celle utilisée par le développeur Web, pour l'utilisateur, l'application sera appelée de la même manière qu'un site internet standard en PHP, à savoir au travers d'un appel à une URL dans un navigateur web.

Le développeur doit prévoir cette interface graphique, car c'est au final la seule partie que verra l'utilisateur final.

Attention, il est à noter que le développeur n'est pas un intégrateur. En effet, celui-ci ne maitrise pas le HTML et les feuilles de styles CSS permettant de réaliser l'interface graphique d'une application Web. En pratique, il sait les utiliser, mais ce n'est normalement pas dans ses attributions.

2.2.6.2. Le graphiste

Le graphiste est la personne qui va s'occuper de tout l'aspect de l'interface graphique, que ce soit les couleurs, la charte graphique, la création du style, ainsi que tout ce qui va rendre l'interface graphique finale attirante et agréable à voir.

Le graphiste, ou designer graphique, a le rôle de dessinateur, et est chargé de la conception des projets graphiques.

Le graphisme est une discipline qui consiste principalement à créer, choisir, utiliser des éléments graphiques, que ce soient dessins, photos, couleurs, etc. pour élaborer un objet de communication, ou bien de culture. Dans notre cas, l'objet de communication est une application web. Ce dernier est un professionnel de la communication qui conçoit des solutions de communication visuelle. Il travaille sur le sens des messages à l'aide des formes

graphiques. Il manie avec perfection les outils de création d'images tels que le très connu Photoshop de la firme Adobe.

Il est à noter ici que le graphique n'a pas en théorie de connaissances techniques sur un projet informatique. Il ne maitrise que les éléments visuels. Il ne doit donc pas avoir à toucher aux algorithmes que conçois le développeur, ni à intégrer les éléments visuels qu'il a créé au projet. C'est le rôle de l'intégrateur que d'intégrer ces éléments.

En pratique, un profil de graphiste possède souvent des compétences en intégration, voire également des compétences, ou des notions, en développement.

2.2.6.3. L'intégrateur

L'intégrateur est en pratique principalement présent dans les structures des agences de communications, basées sur la réalisation de sites internet. On parlera alors ici d'intégrateur Web.

L'intégrateur web est une personne qui va être chargée de produire, traduire puis transposer en langage informatique les maquettes fournies par l'équipe graphique. Son rôle consiste à réaliser l'ensemble de la mise en page d'un site web en y assemblant tous les éléments des maquettes graphiques (textes, images, sons) tout en respectant le cahier des charges. Ce dernier doit s'efforcer de suivre les normes et les standards, afin de rendre son travail accessible et retranscrit parfaitement sur chaque navigateur. Pour mettre en page le site web, l'intégrateur web s'appuie sur le langage HTML et le CSS, avec parfois du JavaScript.

C'est en réalité l'intermédiaire entre le développeur et le graphiste. Il va travailler en collaboration avec les deux autres profils.

Il est à noter qu'il participe également à la qualité du site, tout en respectant les normes d'accessibilité et d'ergonomie.

3. Les éléments à mettre en place ou à améliorer afin d'obtenir une interface graphique idéale

Dans cette troisième et dernière partie de ce mémoire, nous allons nous intéresser aux différents éléments qui pour moi sont importants à garder, ainsi que ceux à améliorer afin de se rapprocher de cette interface graphique idéale au niveau d'une application.

Cette interface graphique idéale se doit de satisfaire tous les profils qui lui sont liés de près ou de loin, c'est-à-dire des différents acteurs ayant à y travailler durant le développement de l'application et plus particulièrement de sa mise en place. Celle-ci doit également satisfaire ses utilisateurs finaux, qui eux vont avoir à l'utiliser de manière plus ou moins récurrente, en répondant aux différents critères relatés plus haut, au niveau de la couche applicative principalement.

Je vais donc donner mon point de vue personnel sur chacun des éléments qui me semblent indispensables ou non d'intégrer à un projet afin que son interface graphique approche la perfection. Je pourrai également avancer des axes d'amélioration sur des outils ou des critères qui me semblent intéressants à intégrer, mais qui aujourd'hui ne sont pas assez aboutis et / ou performants selon moi, et qui nécessitent eux-mêmes d'être améliorés et d'arriver à maturité afin de pouvoir s'en servir correctement.

3.1. Au niveau applicatif

Je pense qu'il est important de parler des différents critères que doit respecter l'application.

Il est également nécessaire pour moi de discuter des différents outils et applications allant permettre à ces critères d'être respectés.

En outre, il serait judicieux de déterminer quelles seraient les différents paramètres qui pourraient être améliorés.

L'accessibilité dans une application est nécessaire afin de pouvoir être clairement visitée et utilisée sans problème, quel que soit le profil de la personne qui la visite.

En effet, les personnes possédant un handicap ont tout à fait le droit d'avoir accès à l'ensemble des fonctionnalités d'une application, au même titre que les personnes valides.

Je trouve donc tout à fait normal qu'une application considérée comme publique aient des obligations légales de répondre à un ensemble de critères afin d'être considérées comme accessibles.

De plus, j'estime que malheureusement, il reste encore beaucoup de travail à réaliser avant que la majeure partie des applications publiques soient accessibles.

En effet, bien qu'en Europe et en France, des lois obligent une certaine accessibilité, ce n'est pas forcément le cas dans les autres pays et zones géographiques.

En outre, ces obligations légales ne sont pas assez respectées par la plupart des applications et / ou sites internet.

Mais encore une fois, à qui remettre la faute ? En effet, si l'on prend un exemple d'application commandée chez une SSII, va – t –on blâmer le client, qui ne l'a pas commandé dans le cahier des charges, ou le prestataire, qui ne met pas en place des structures allant permettre l'accessibilité ?

De la même manière, les particuliers, n'ayant que très peu de connaissances techniques, qui créent un site web, ne vont généralement même pas penser à cet aspect.

Il y a clairement un manque de sensibilisation à ce niveau, et je pense que c'est sur ce point qu'il faudrait commencer à communiquer.

Le critère d'accessibilité est donc pour moi essentiel à une interface idéale.

Nous allons maintenant parler du responsive. Le responsive s'adresse principalement aux sites internet et applications web disponibles par navigateur, du fait de la grande tendance actuelle à développer des applications web.

Pour rappel, le responsive signifie de posséder une interface qui sera toujours lisible et compréhensive, et ce même lorsque les dimensions de l'application sont modifiées.

Je pense pour ma part que le responsive est une excellente idée.

Au vu de l'explosion ces dernières années des Smarthphones et tablettes, les habitudes de navigation ont changé. La plupart des personnes naviguent maintenant à partir de leur terminal tactile et non plus sur PC.

La différence de structure est importante entre les terminaux tactiles, généralement petits, et dont l'interaction s'effectue avec les doigts, avec les terminaux plus classiques comme les PC avec clavier et souris.

Dans un premier temps, face à la demande, ce sont des applications client lourd qui ont été développées. Seulement, leur cout de développement est élevé, et la multiplication des systèmes d'exploitation mobiles n'a pas aidé. En effet, pour avoir une visibilité maximale, il fallait une application Android, une Ios, une windows phone, une BADA, etc.

Ceci était bien trop couteux. C'est pour cela que je trouve judicieux d'avoir créé des outils permettant avec aisance de créer des applications responsives, consultables aussi bien sur PC classique que sur terminaux tactiles.

Il est certes vrai qu'une application native sera toujours plus performante sur terminal tactile qu'une application web, mais son cout et sa maintenance, une seule version du site, demandent bien moins de d'investissement.

Ayant utilisé BootStrap de Twitter, j'ai été bluffé par son efficacité.

Je pense sincèrement que le responsive est en train de rentrer dans les habitudes de développement, et que d'ici peu, la majorité des applications web seront responsive.

Ce critère est donc essentiel à une interface idéale.

Voyons maintenant le critère de testabilité de l'interface d'application.

Par testable, j'entends en effet tests automatisés. En effet, sinon le test non automatisé en soit n'est pas compliqué, cependant les répétitions que demandent l'exécution des cas de test peuvent vite être source d'erreur, et les tests vont laisser passer des anomalies.

Les tests automatisés n'ont pas le même objectif que les critères précédents.

En effet, ces derniers sont vraiment des critères qui vont aider l'utilisateur final à avoir une meilleure expérience utilisateur au travers de l'interface graphique.

Ici, le critère est plutôt adressé à la phase de développement de l'application.

Elle va surtout permettre d'éviter la livraison au client d'une interface graphique boguée et / ou incomplète.

Les tests automatisés sont réalisés par une application externe spécialisée.

Ces applications peuvent se baser sur des tests purement graphiques, en cliquant sur une coordonnée précisée dans le cas de test par exemple. D'autres se basent sur le code disponible qui permet l'affichage de l'interface. C'est le cas des langages HTML / CSS qui permettent d'afficher l'interface graphique d'une application web dans un navigateur.

Ces logiciels de tests vont alors se baser sur des identifiants précis. L'avantage ici est que si des composants changent de position, cela n'influera pas sur les tests.

Le logiciel Selenium permet entre autre ce genre de test. Je le trouve très performant, l'utilisant en entreprise.

Cependant, celui-ci est limité, car il ne peut s'utiliser que sur des interfaces graphiques web.

De plus, j'ai déjà eu un cas où Selenium n'était pas utilisable. En effet, lorsque l'on utilise des framework graphiques pour nos applications JEE, le développeur ne maitrise pas tout le temps l'intégralité de l'interface graphique.

En effet, je prends l'exemple de GWT, qui signifie Google Web Toolkit. Sa force selon moi est dans le fait que le développeur n'a pas à développer la structure en HTML / CSS de l'interface graphique.

Celui-ci va générer automatiquement ce code HTML / CSS à partir du code Java développé pour l'interface graphique.

Dans ce cas précis, ne maitrisant pas le code final de l'interface graphique, nous n'avons pu utiliser Selenium.

Pour conclure sur ce critère, je dirais que les tests automatisés sont très pratiques pour les développeurs. Les outils actuels nécessiteraient cependant une petite amélioration selon moi.

Ce critère étant essentiel pour vérifier le fonctionnement normal de l'interface, ainsi que la non régression, il peut être validé comme essentiel à une interface idéale.

Intéressons-nous maintenant à la personnalisation de l'interface graphique.

Il faut bien garder à l'esprit que l'interface idéale serait l'interface qui plairait à tous.

Or, tout le monde ne possède pas les mêmes gouts.

La personnalisation est aujourd'hui un élément important lorsque l'on développe pour le plus grand nombre.

J'estime que l'on peut observer plusieurs types de personnalisation.

La première concerne uniquement les graphismes et la charte graphique, en donnant la possibilité à l'utilisateur de changer de les modifier en fonction de ses gouts. On peut envisager pour cela la mise à disposition de plusieurs thèmes de base.

La seconde personnalisation est bien plus importante, et ne se limite pas aux couleurs.

Il ne faut pas oublier que chacun et diffèrent. De ce fait, chacun possède une logique de fonctionnement qui lui est propre.

C'est pour cela que cette deuxième personnalisation va concerner l'ensemble de l'interface graphique. L'utilisateur pourrait de cette manière paramétrer ses préférences, et alors obtenir une interface unique qui lui est propre.

Prenons exemple de Gmail, qui selon moi fonctionne dans ce sens. L'application de messagerie de Google permet bien de choisir un thème d'arrière-plan, qui conforte le

premier type de personnalisation. Mais de plus, il permet à chacun de paramétrer en conséquence son interface graphique.

En effet, il nous est possible de créer des catégories de mail, où les mails seront rangés directement dans la catégorie qui lui est associée.

De plus, les composants peuvent être affichés ou cachés en fonction des gouts. Ils peuvent également être déplacés.

C'est le cas parfait d'une application personnalisable selon moi. Et en effet, ce critère est selon moi nécessaire à l'interface utilisateur idéale.

J'aimerais également parler d'un critère que devrait présenter l'interface graphique idéale : l'intuitivité.

En effet, selon moi, l'interface sera idéale à partir du moment où elle sera intuitive pour l'utilisateur.

Car il ne faut pas oublier que c'est l'objectif de toute interface graphique : pouvoir communiquer de la manière la plus intuitive possible avec un programme.

En effet, ce critère regroupe en majeur partie les autres dont j'ai parlé plus haut.

Car une interface intuitive va être une interface personnalisable par l'utilisateur, qui l'aura créé de manière à ce qu'elle soit la plus logique selon lui.

De la même manière, une interface intuitive sera une interface qui nécessitera d'être responsive, car claire quelle que soit la taille de l'affichage.

Enfin, une interface intuitive sera forcément accessible à tous.

3.2. Au niveau Présentation

Au niveau de la couche présentation, qui va regrouper les bonnes pratiques au niveau du code de l'application, je suis satisfait de l'efficacité des différents patrons de conception existants présentés plus haut.

Rappelons rapidement qui ils sont et le rôle qu'ils jouent.

Le design pattern Modèle-Vue-Contrôleur sépare les données, l'interface homme-machine et la logique de contrôle. C'est un modèle de conception imposant donc une séparation en 3 couches distinctes.

Le patron de conception Modèle-Vue-Présentateur dérive du patron Modèle-Vue-Contrôleur. Ici, le présentateur prend la place du contrôleur, et obtient le rôle d'intermédiaire entre le modèle et la vue.

Le patron de conception Modèle-Vue-VueModèle est similaire aux patrons précédents, et a été développé par Microsoft. Son but est de proposer une méthode de conception adapté à ses propres outils de développement. Microsoft s'est fortement inspiré du patron de conception MVP pour réaliser celui-ci.

Chacun d'eux permet vraiment de distinguer la partie du code métier à la partie s'occupant de l'interface graphique.

J'estime cependant que ce n'est pas suffisant. Le développeur est toujours amené à créer cette interface et à l'agencer, alors que selon moi, ce n'est pas de son ressort. En effet, les compétences de développeur ne devraient pas contenir cette tâche que comporte le fait de mettre en page l'interface graphique.

Il est possible qu'en théorie, ce rôle soit bien associé à un domaine de compétence que n'inclut pas le développeur, mais en pratique, j'ai toujours vu ce dernier avoir la charge de la réaliser.

De plus, je préconise l'utilisation de langages de programmation qui vont permettre à un profil pas spécialement technique de construire l'interface graphique finale.

En effet, certains langages permettent d'externaliser dans des fichiers de configuration, tels que des fichiers de type XML, ou bien d'autres manières.

Je prends exemple sur le langage Java et le développement d'applications sur les smartphones ou tablettes fonctionnant sous Android.

Du fait de leur grande diversité de résolution d'écran, et donc d'affichages possibles pour l'application, l'interface peut être réalisée en XML avec des balises et suivant une certaine hiérarchie.

En effet, certains composants seront alors relatifs à d'autres, et l'affichage d'adapte alors automatiquement pour répondre à la structure.

Un autre gros avantage à cette externalisation de la construction de l'interface graphique par des fichiers tels que XML est à mon sens la possibilité offerte aux profils non techniques de pouvoir manipuler les composants graphique au sein de cette interface.

En effet, une personne, par exemple un graphiste, ou bien un intégrateur, n'aura besoin que de s'habituer au fonctionnement par balise du XML, et pourra bien plus facilement travailler sur cette interface graphique, que s'il devait rentrer dans le code source du programme en question.

Toujours concernant la création d'une interface graphique, je pense qu'aujourd'hui le secteur de l'informatique tient un problème majeur.

En effet, il arrive bien trop souvent à mon gout que les choix de construction et des différents agencements des composants d'interface graphique soient délaissés et qu'au final il incombe au développeur de réaliser cette tâche.

Le problème ici est que ce développeur, qui n'a pas eu de formation en ergonomie, n'a pas les compétences pour réaliser une interface qui sera intuitive et réellement adaptée aux besoins.

Par la force des choses, il effectuera un travail correct, mais qui ne pourra bien évidemment pas rivaliser avec une tâche effectuée par un expert dans le domaine.

Je tiens à préciser que c'est juste un constat. J'ai moi-même du réaliser divers impacts au niveau de l'interface graphique qui laissaient le client très content du résultat.

Mais je suis tout à fait conscient que quelque chose de mieux pourrait être fait.

3.3. Au niveau du système d'exploitation

Il m'est très difficile de déterminer au niveau de l'environnement graphique les éléments les plus importants, tant ceux-ci sont complexes.

Mais ce que je peux dire, c'est que si l'on compare X Window à Win32, l'avantage revient selon moi à l'environnement graphique des systèmes Linux / UNIX.

En effet, ce dernier à divers avantages vis-à-vis de Win32 qui me font le préférer.

Tout d'abord, cette utilisation du protocole X et par conséquent du modèle client – server est vraiment intéressant et promet de nombreuses possibilités d'utilisation.

En outre, le fait qu'il permette de choisir les différents composants qui l'intègrent, que sont le système de fenêtrage, le gestionnaire de fenêtres, l'environnement graphique et le gestionnaire de fichiers est très intéressant.

J'aime personnellement pouvoir avoir le choix lorsque je dois adopter quelque chose.

Cela joue peut être avec mes préférences personnelles, mais pour ces raisons, je préfère X11.

Cependant, je ne dis pas que celui –ci est parfait, loin de là.

Ce que je peux tout de même concéder à ces environnements de bureaux, c'est qu'ils permettent tout de même que l'on surcharge leur boite à outils de composants par défaut.

Je pense qu'à l'avenir, de plus en plus d'éléments contenus dans ces environnements auront la possibilité d'être soit modifiés, soit surchargés par d'autres concepts externes.

En ce sens, X Window est tout de même bien avancé.

Conclusion

Suite à cette partie de réflexion personnelle sur les éléments à prendre en compte afin d'approcher une interface graphique parfaite, nous pouvons conclure sur le sujet de ce mémoire.

Une interface graphique est, comme on l'a constaté tout au long de ce mémoire, une partie à part entière d'un projet informatique. Celle-ci sera l'unique partie du projet réalisé qui sera visible par les utilisateurs finaux. Ce n'est donc pas une partie à négliger.

Une interface graphique possède une structure qui aujourd'hui a relativement évolué par rapport à ses débuts dans les années 70 avec la sortie de la première interface graphique, qui ne comprenait que peu de composants d'interface graphique, ainsi qu'une possibilité relativement faible d'interaction avec l'utilisateur.

Cette interface, révolutionnaire pour l'époque, est due au laboratoire de recherche Xerox PARC avec le premier ordinateur à interface graphique nommé Alto

Cet ordinateur est muni initialement de deux périphériques d'entrée, à savoir le clavier, qui va permettre de saisir des données sur cette interface, et un périphérique de pointage au travers de la souris.

Très vite, de nombreuses sociétés d'électronique et d'informatique se ruent sur ce nouveau marché prometteur. C'est le cas notamment des firmes Apple, IBM et Microsoft. Apple fut la première société à avoir toujours un train d'avance sur le marché, notamment grâce à son interface graphique bien plus aboutie.

Microsoft reviendra dans la course en 1995 avec son système d'exploitation Windows 95.

Une alternative à ce duopole se développe également, Linux. Issu d'Unix, et par conséquent du monde des serveurs, Linux va se doter d'interfaces utilisateurs plus compatibles avec les besoins du grand public.

Aujourd'hui, les deux structures principales de ces interfaces graphiques, appelées environnement graphique, découlent de ces avancées historiques au sein de chaque

système d'exploitation. La première, X Window, est contenu dans l'ensemble des systèmes d'exploitation et distributions basées sur Linux / UNIX. Le second environnement graphique est celui de Microsoft, disponible depuis Windows 95, et se nomme Win32, ou Windows API.

Ces deux environnements graphiques possèdent un fonctionnement global similaire, bien qu'ils soient différents au niveau de leur architecture interne. Ainsi, leur structure globale peut se diviser en plusieurs parties distinctes, qui sont :

- La couche applicative, qui se situe au niveau le plus haut, et va correspondre au niveau du langage informatique principalement.
- La couche comprenant le widget toolkit ainsi que le les différents widgets.
- La couche comprenant la structure principale de traitement bas niveau des différents éléments graphiques, regroupant l'environnement graphique de chaque système d'exploitation, c'est-à-dire soit X Window pour les systèmes d'exploitation compatibles Linux / UNIX, soit Windows API pour les systèmes d'exploitations Windows
- La couche la plus basse au niveau logiciel, qui correspond au pilote graphique. C'est celui-ci qui va permettre le dialogue entre la couche précédente et le matériel de sortie, à savoir le périphérique de retour image tel que l'écran

Les toolkits intègrent un certain de composants graphiques, chacun ayant un visuel ainsi que des propriétés qui lui sont propres. Ces composants peuvent être catégorisés dans plusieurs familles que sont les éléments d'affichage simple, les boutons, les menus, les conteneurs, les listes, les champs utilisateur, les aides au retour utilisateur, ainsi que les fenêtres.

Il faut savoir que chaque environnement de bureau en possède au moins un toolkit de base, mais aujourd'hui, la plupart des applications utilisent des widget toolkit alternatifs, qu'elles soient dites client lourd ou accessibles via navigateur.

L'environnement graphique X window apporte deux spécificités principales, que sont sa structure de dialogue interne fonctionnant en modèle client-serveur, et sa structure dite souple allant permettre à l'utilisateur de personnaliser entièrement les différents

composants de cet environnement graphique, que sont le système de fenêtrage, le gestionnaire de fenêtres, et le gestionnaire de fichiers, qui sont chacun interchangeables et modifiables.

L'environnement graphique Win32 comprend en réalité un ensemble normalisé de fonctions allant permettre aux logiciels d'utiliser les fonctionnalités de ce système d'exploitation. C'est GDI qui possède l'ensemble de fonctions composant l'environnement graphique. Ce dernier va pouvoir représenter des éléments graphiques d'une part, mais va également permettre leur transmission aux périphériques de sortie d'autre part.

Cependant, c'est aux fonctions comprises dans le composant User de Win32 que se chargeront de l'affichage des fenêtres, des menus et autres contenus graphiques.

GDI va également permettre d'imprimer des images au travers d'un type d'imprimante particulier. Ces imprimantes compatibles, également appelées GDI, sont des périphériques externes qui ne possèdent ni RAM, ni CPU, et très peu d'électronique, en comparaison à une imprimante plutôt standard. Son fonctionnement est simple : l'image d'origine est transformée en format bitmap, puis simplement envoyée telle quelle à l'imprimante, qui édite le document.

Maintenant que l'on a déterminé les structures des différents environnements graphiques, passons aux objectifs auquel doit répondre l'interface graphique.

Au niveau de la couche présentation, intéressons-nous aux design patterns. Une application doit mettre en place des structures allant permettre de la gestion de manière optimale les interactions entre l'interface graphique et le traitement des informations afin de proposer une performance maximale. Nous avons traité 3 design patterns ici.

Le design pattern Modèle-Vue-Contrôleur qui est un pattern architectural qui sépare les données, l'interface homme-machine et la logique de contrôle. C'est un modèle de conception imposant donc une séparation en 3 couches distinctes.

Le patron de conception Modèle-Vue-Présentateur est un modèle dérivé du patron Modèle-Vue-Contrôleur. Il reprend la structure du modèle MVC à quelques différences près. En effet,

le présentateur prend la place du contrôleur, et obtient le rôle d'intermédiaire entre le modèle et la vue.

Le patron de conception Modèle-Vue-VueModèle est un patron similaire aux patrons précédents, qui a été développé par Microsoft. Son but est de proposer une méthode de conception adapté à ses propres outils de développement. Microsoft s'est fortement inspiré du patron de conception MVP pour réaliser celui-ci.

De plus, dans un projet En effet, une interface graphique doit répondre à différents critères :

D'une part son accessibilité par la plus grande partie de son public, que ce soit au niveau du support sur lequel on consulte l'application, ou au niveau des différences entre les profils de personnes qui visitent, ou utilisent l'application, web ou non. Ces personnes peuvent être malvoyantes, ou atteint d'une surdité. Tant d'éléments qu'il faut prendre en compte lors de la réalisation d'une application.

D'autre part, sa propriété responsive, afin de s'adapter et d'être visible, disponible et compréhensible à l'ensemble des terminaux et des résolutions d'écrans existants.

En outre, elle doit répondre au critère de testabilité automatisée, afin d'obtenir constamment une application performante et sans bugs ni anomalies.

De plus, elle doit permettre à ses utilisateurs de pouvoir personnaliser leur interface à leur convenance, par la mise à disposition de thèmes ou de la modification de la charte graphique par exemple.

En outre, l'application, et plus particulièrement son interface graphique, doit être évolutive. Cela signifie qu'elle peut être modifiée sans avoir un impact sur le reste de l'application.

Enfin, elle doit également être modifiable par l'ensemble des corps de métiers ayant à travailler au sein du projet, qu'ils soient techniques ou non.

Il faut ajouter que l'interface graphique est le seul élément d'un projet informatique faisant cohabiter plusieurs domaines de compétence, et donc de métiers.

Un développeur est un informaticien qui réalise des logiciels ou des applications destinés à paraître sur Internet ou sur client lourd, en créant des algorithmes et en les mettant en œuvre dans un langage de programmation prédéfini.

Le graphiste est la personne qui va s'occuper de tout l'aspect de l'interface graphique, que ce soit les couleurs, la charte graphique, la création du style, ainsi que tout ce qui va rendre l'interface graphique finale attirante et agréable à voir.

L'intégrateur web est une personne qui va être chargée de produire, traduire puis transposer en langage informatique les maquettes fournies par l'équipe graphique. Son rôle consiste à réaliser l'ensemble de la mise en page d'un site web en y assemblant tous les éléments des maquettes graphiques tout en respectant le cahier des charges.

Voyons maintenant ce que sont selon moi les éléments à garder et à améliorer pour chaque couche de l'interface graphique.

Tout d'abord, je pense que l'ensemble des critères étudiés sont nécessaires à l'obtention de l'interface graphique idéale.

Comme je l'affirme plus haut, chacun joue son rôle dans cette réalisation.

Ils peuvent être considérés comme complémentaires les uns des autres.

En effet, j'insiste sur le fait que selon moi, l'interface graphique idéale est une interface graphique qui tout simplement intuitive pour son utilisateur.

A partir de là, l'accessibilité va permettre à tous de comprendre son contenu, même les personnes handicapées.

Le responsive, quant à lui, permettra la lisibilité, et ainsi permettre la compréhension du contenu sur l'ensemble des terminaux possibles.

La personnalisation permettra, lui, à chacun d'entre nous de personnaliser son interface qui lui correspond le mieux, et ainsi, son interface idéale, car n'oublions pas que quelque chose d'idéal est subjectif.

Cependant, les différentes préconisations que j'ai effectuées s'adressent uniquement à un avenir proche. En effet, avec le développement constant des objets électroniques, s'adaptent les interactions Homme Machine également.

En effet, il ne faut pas perdre de vue de la plupart des interfaces actuelles ont été imaginées par les pionniers de l'interaction homme–machine. Elles sont donc adaptées aux premiers périphériques que sont le clavier et la souris.

Toute la difficulté est de rendre des interactions initialement complexes plutôt simples et naturelles. Que ce soit capture du mouvement, table interactive, reconnaissance vocale, etc. Le développement d'interfaces innovantes va nécessiter de replacement celle qui est au cœur du processus de conception, afin d'anticiper les usages.

Ces interfaces vont devoir s'adapter à nos capacités physiques, cognitives et sociales. Cependant, la plupart n'exploitent pas toutes les potentialités de la communication humaine.

Par exemple, les agents virtuels autonomes sont capables de dialoguer, verbalement et non-verbalement, avec un utilisateur en mimant expressions du visage, intonations de la voix et gestes communicatifs humains. Si l'illusion d'un échange est impressionnant, pourrons-nous bientôt discuter réellement avec une machine ?

Aujourd'hui, l'évolution des interfaces tend vers la dématérialisation de l'interaction. Commander par la pensée est désormais possible, en traduisant l'activité électrique du cerveau en actions sur un ordinateur.

Ces dispositifs d'interface cerveau-machine représentent également un réel espoir pour les personnes paralysées, et ouvriront peut-être un jour de nouvelles perspectives de communication « télépathique », telles que l'on peut observer dans les films de science-fiction.

Tables des matières

Bibliographie

Historique de l'interface graphique : http://pro.01net.com/

Patterns:

- MVC : http://developpez.com/
- MVVM : http://msdn.microsoft.com/

Responsive : http://designspartan.com/

Graphiste : http://metiers.internet.gouv.fr/metier/integrateur-web

Glossaire

GUI : terme anglais signifiant « interface graphique »

OS : terme anglais signifiant système d'exploitation

Pattern : diminutif voulant dire Design pattern

Design pattern : signifie Patron de conception en français

Annexes

Résumé d'anglais

The document I have to summarize is my end study's memory. It is an 80pages long document which the subject is about the GUI, for Graphical User Interface. In this we can see an introduction of the subject, followed by three main distinguishing parts. At last, we have a conclusion of all of that.

The first deals with the Graphical User Interface's structure.

The second is about the different goals the GUI has to reach in order to be acceptable.

The third part is my point of view about which criteria should be in a component or in another part of the Graphical User Interface's structure.

Now let's begin the summary by entering deeply in the subject.

The introduction describes first the Graphical User Interface. Graphical User Interface is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators. Graphical User Interfaces were introduced in order to come after the command-line interfaces which require commands to be typed on the keyboard.

The first device using Graphical User Interfaces was the Xerox PARC in 1981.

Then the first computer using a Graphical User Interface and a mouse was the Apple in 1983.

Many firms, like IBM or Microsoft, started to create their own Graphical User Interface.

But for a while, until 1995, Apple had the most advanced Graphical User Interface. At this time, IBM stopped his research on his own Graphical User Interface, developed conjointly with Microsoft, called the OS/2.

Microsoft developed a Graphical User Interface called Windows, but in the first time, it wasn't a fully Graphical User Interface.

Hopefully for him, in 1995, the last Microsoft's operating system named Windows 95, had a real success.

It was the first time a Windows version was not a only a Graphical component using the MS-DOS operating system, but a real complete operating system itself.

Now we can find two main Graphical User Interfaces structures. The first is based on Microsoft's operating system, and the other on Linux / UNIX based on operating system. The first structure is called Win32, while the second is X Window.

Each Graphical User Interface global structure contains a widget toolkit. This is a toolkit which includes many widgets.

The different widgets have much functionality.

For example, the label is a text zone in which some text will inform you about something.

The text Field, him, allows the user to tap some text into him, in order to apply later the form containing the field, and sends information on the server distant application.

These two structures have both unique functioning.

On the one hand, X Window is using a client-server model pour his internal communication between sub-elements it contains, while Win32 is using standard communication.

On the other hand, Win32 contains an element called GDI, meaning Graphical Device Interface. This element permits to print easily any image or picture directly on a compatible printer.

Let’s now deal with the different design patterns existing which allow to the source code’s project a particular structure. With these structures, Graphical User Interfaces code and other code would not be mixed, and as a consequence, easier to maintain.

These three design pattern are the MCV for Model–view–controller, the MVP for Model–view–presenter, and the MVVM for Model – View - ViewModel.

Each one of them shows us a different way of separating the view of the model code.

With the Model–view–controller pattern, the controller is the one which control the view, and send requests to the model. This one answer with data, that the view takes, and updates his fields.

With the Model–view–presenter pattern, the presenter will take place between the model and the view. It’s the only component that can dialogue with the others. The view can’t directly ask information to the model, and in the same way, the model can’t send directly the data asked. They both have to dialogue with the presenter, which will transfer the information.

We can now see the different points a Graphical User Interface have to reach in order to be valuable.

First, the Graphical User Interface has to be responsive. It means that with any electronic device, a program have to be nice to see, without any graphical bug.

The program has to reach another goal: his Graphical User Interface has to be testable by another test-program, like Selenium for example. This application will execute previous saved tests on the Graphical User Interfaces original program. This permits to be certain about the Graphical User Interfaces application quality.

The Graphical User Interface should be easy to personalize by any non-technical person. In fact, today, many ready to use websites can be found on the net. Sadly, the default Graphical User Interface is often no really beautiful. In this case, the application has to allow the final user to modify this default Graphical User Interface.

Moreover, today, everyone should have an access to a web application, and understands all information he can find on it. By meaning everyone, I include even the handicapped ones.

So when we publish a video, we have to join subtitle, for the deaf persons.

Furthermore, a blind person should find an oral descriptor when she consults an image or a graphic.

We have to precise that the Graphical User Interface is the only project's part that gathers different jobs on it.

We have first the developer, which make the source code of the program.

The second job is the designer. This one creates some designs, images, graphics standards, and others graphical elements the project may need.

The third job involved in this is the integrator. His goal is to integrate graphical content onto the application's source code.

This job can be viewed as the link between the two others jobs explained.

In the last part, I'm speaking about for each element composing a Graphical User Interface, what should be remained in order to create the utopic ideal Graphical User Interface.

First, I think that the Graphical User Interfaces future is on the internet, because an application we can access with a simple navigator is better than a program we can only access on only one operating system.

Moreover, I think that a tool should be developed in order to help developer to integrate the accessibility easier in a web site, or a web application.

Furthermore, as for me, a new design pattern should be invented, in order to separate more and more the code source used for the interface, and the source code for the application's functionalities.

Concerning the responsive point, I think that it is one of the most awesome tools recently developed.

I hope that other tools like this one will appear in the future.

As for me, the different jobs working on the Graphical User Interface should have a better communication, and find a way to work hand by hand to approach the best.

To conclude, I want to say that the Graphical User Interface is, on a project, a real complex object which cannot be left for another part of the application.

In fact, the Graphical User Interface is the only thing the user will remain of the whole project you made, because this is the only visible part of the project.