UNIVERSITE CATHOLIQUE DE LOUVAIN

FACULTE DES SCIENCES APPLIQUEES

DÉPARTEMENT D'INGÉNIERIE INFORMATIQUE

Etude de la validation ergonomique d'une interface homme-machine à la conception

Promoteur : Professeur J. VANDERDONCKT

Mémoire présenté en vue
de l'obtention du grade
de licencié en informatique

par
David DE LIMELETTE

Louvain-La-Neuve
Année académique 2004-2005

À ma grand-mère Nany, décédée en mars 2005,
et à ma tante Suzette, décédée en avril 2005.

Remerciements

Je tiens à remercier les personnes suivantes pour leur contribution dans la réalisation de ce mémoire.

Monsieur J. Vanderdonckt, pour la supervision de mon mémoire, pour avoir toujours proposé de nouvelles idées et partagé son enthousiasme.
Mon oncle Jacques Ponteville, pour les relectures, les corrections et ses conseils.
Mes parents pour leur soutien, depuis toujours, dans mes études.
Ma fiancée Xingzhu, pour sa présence près de moi dans les moments difficiles et sa participation dans ce mémoire.

Tables des matières

Chapitre 1 Introduction

Il existe beaucoup d'outils de validation automatique de règles, que ce soit des règles d'ergonomie ou d'accessibilité. L'ergonomie et l'accessibilité sont des propriétés des interfaces utilisateurs qui sont souvent de plus en plus appréciées.

En regardant de plus près ce que ces termes impliquent, nous comprendrons l'intérêt et l'importance de rendre les interfaces ergonomiques et accessibles.

Dans la section 1.1, nous présenterons l'ergonomie dans sa définition, son objectif et son application dans le domaine informatique. La section 1.2 sera consacrée à l'accessibilité, sa définition, son objectif, son application dans les interfaces homme-machine et les aides techniques utilisées par les utilisateurs handicapés. La section 1.3 présentera ce que l'on entend par règle d'ergonomie et d'accessibilité. Nous présenterons ce que l'on entend par validation automatique dans la section 1.4, et ce que l'on entend par phase de conception dans la section 1.5. Finalement nous présenterons la méthodologie de notre étude dans la section 1.6.

A partir de notre méthodologie, nous observerons l'état de l'art dans le chapitre suivant.

1.1 Ergonomie

Afin de saisir l'importance de l'ergonomie par l'absurde, nous présentons quelques exemples de mauvaise conception d'interface. En voici deux, repris du site « User Interface Hall of Shame » [Mar05].

- **Premier exemple**

Dans le logiciel WinRental, nous pouvons voir un mécanisme de sélection de date qui est composé de 3 zones de contrôle, pour le jour, le mois et l'année. Comme vous le voyez dans la prise d'écran suivante, il est même possible de créer sa propre date.

De plus, lorsqu'il y a une valeur "Null" dans la base de données, le champ de l'année indique zéro.

Ce choix n'est vraiment pas ergonomique, d'autant plus qu'il existe un élément d'écran prévu à cette effet : "Date Time Picker". Son utilisation serait nettement plus appropriée.

- **Deuxième exemple**

Il est également question ici d'un message d'erreur (voir la Figure 1). Celui-ci appartient à l'outil de contrôle de version « ClearCase » du logiciel IBM Rational Software. Cela prouve bien que même les grands noms du logiciel ne sont pas à l'abri d'erreur d'ergonomie.

En effet, ce message d'erreur ne respecte pas le critère de brièveté, l'un des critères d'ergonomie. Idéalement, ce message d'erreur devrait respecter la règle d'ergonomie suivante issue du livre de J-F Nogier [Nog02]

> *Le message d'erreur doit préciser la nature du problème et donner les moyens d'y remédier.*
>
> *Produire des messages brefs, concis et pertinents.*

Mais là n'est pas la seule erreur, puisque les deux boutons « ok » et « cancel » effectuent exactement la même opération. Ils ferment, tous les deux, le message d'erreur, uniquement.

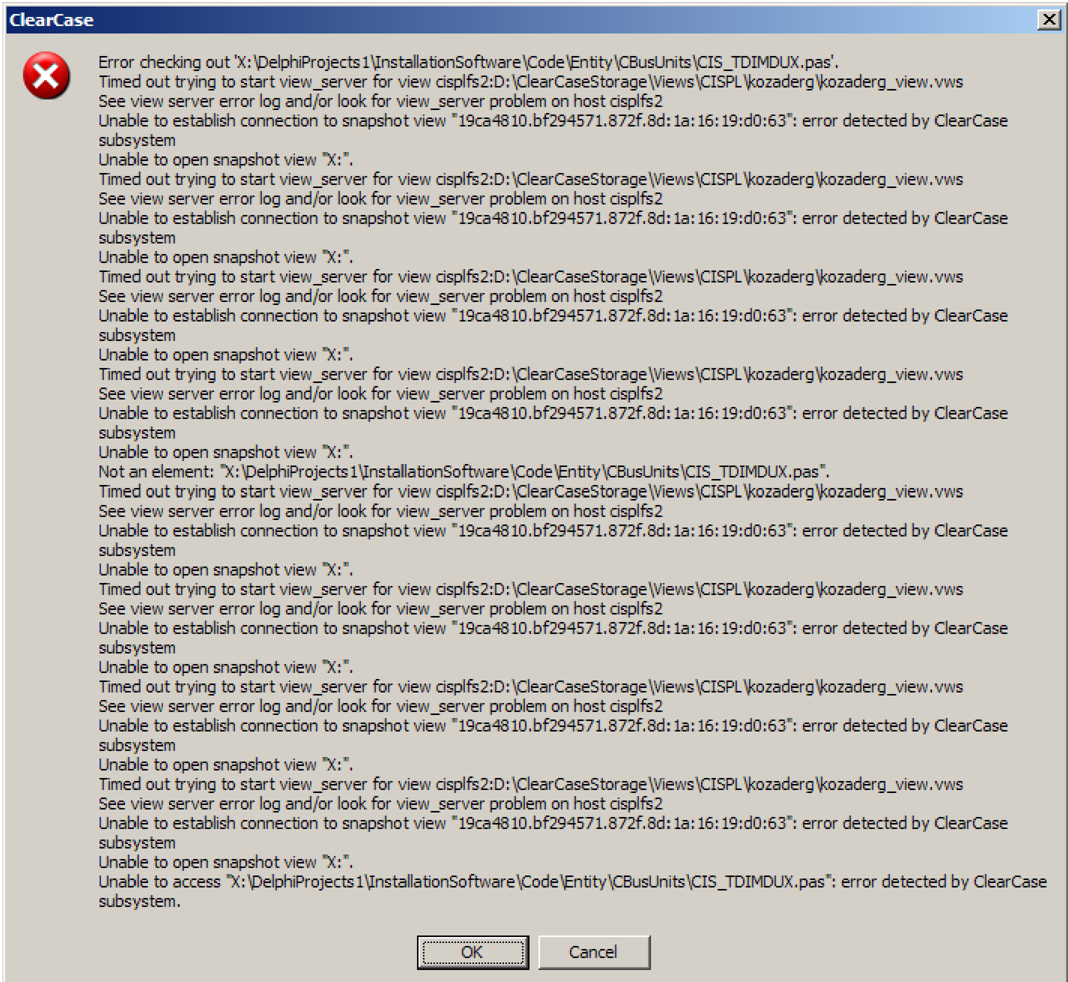


Figure 1 - Message d'erreur du logiciel ClearCase

1.1.1 Définition

Nous allons tenter de définir l'ergonomie à partir des définitions que nous propose Google [Goo05]. En voici deux différentes :

1. The scientific study of human work. The term comes from the Greek words "ergos" meaning work, and "nomos," meaning natural laws of. Ergonomics considers the physical and mental capabilities and limits of the worker as he or she interacts with tools, equipment, work methods, tasks, and the working environment.
   www.lni.wa.gov/wisha/ergo/veg/vegglsry.htm

2. A technical field concerned with optimizing the interface between humans and technology. The field has numerous specialties, including industrial safety and hygiene, human–computer interface design, and the design of control panels in manufacturing plants, cars, airplanes, etc.
   highered.mcgraw-hill.com/sites/0072322098/student_view0/glossary_e-f.html

Nous avons aussi trouvé une autre définition permettant de comprendre ce terme.

3. L'ergonomie (ou l'étude des facteurs humains) est la discipline scientifique qui vise la compréhension fondamentale des interactions entre les êtres humains et les autres composantes d'un système et la mise en œuvre dans la conception de théories, de principes, de méthodes et de données pertinentes afin d'améliorer le bien-être des hommes et l'efficacité globale des systèmes. [IEA05]

Au regard de ces définitions, nous pouvons dire que l'ergonomie est l'étude scientifique des conditions de travail et des relations entre l'homme et la machine. L'ergonomie (ou l'étude des facteurs humains) vise à améliorer le bien-être des hommes et l'efficacité globale des systèmes. Dans notre situation, puisque nous nous intéressons aux interfaces utilisateurs, le terme « ergonomie » concerne une branche de l'ergonomie que nous appelons « Ergonomie d'interface homme-machine ». Nous pouvons dès lors garder en tête la signification suivante :

Rendre une interface ergonomique, c'est concevoir l'interface utilisateur en vue d'améliorer, d'optimiser l'usage de celle-ci par l'utilisateur et ainsi accroître la performance de l'utilisateur.

D'une manière plus simple, il s'agit de faciliter la compréhension et l'usage des interfaces utilisateurs. On parle parfois aussi d' « utilisabilité [1] » d'une interface. L'utilisabilité peut être définie (ISO9241) comme l'efficacité, l'efficience et la satisfaction avec lesquels un utilisateur spécifié réalise un but spécifique dans l'environnement particulier. Où :

- *Efficacité* signifie la précision et la complétude avec lesquelles l'utilisateur peut réaliser le but spécifique dans l'environnement particulier,
- *Efficience* signifie les ressources utilisées en relation avec la précision et la complétude du but réalisé, et
- *Satisfaction* signifie le confort et l'acceptabilité du système de travail pour son utilisateur et les autres personnes affectées par son usage.

1.1.2 Objectif

L'utilisabilité d'une interface est essentielle, car elle détermine la performance de l'utilisateur. Plus l'interface est ergonomique, facile à utiliser, plus l'utilisateur réalisera sa tâche rapidement, sans perte de temps et avec moins de stress. Plusieurs études ont été réalisées et ont prouvé qu'une interface bien conçue permet de diminuer le temps nécessaire à la réalisation de la tâche, de diminuer le taux d'erreur, de réduire le temps nécessaire à la prise de décision, d'augmenter le temps d'extraction de l'information de l'interface, etc. et, par là, d'économiser de l'argent. L'étude de Cope et Uliano [Cop95] a montré qu'une seule fenêtre graphique, rendue plus ergonomique, a augmenté son efficacité et permis à une compagnie d'économiser jusqu'à 20.000 dollars sur un an d'opération.

De plus, l'ergonomie ou l'utilisabilité est un critère de qualité important. Prenons l'exemple d'un site web non ergonomique : lorsqu'un utilisateur arrive sur cette page web et qu'il éprouve de la difficulté à l'utiliser, il y a peu de chance qu'il revienne et il

[1] Ce mot vient de l'anglais « usability ».

gardera une mauvaise image du site web. Le site web en question manque son objectif. Par contre, si la navigation, l'usage de l'interface est clair et précis, s'il arrive rapidement à l'information qu'il recherche, alors il la réutilisera. Un utilisateur moyen[2] sera prêt à faire des concessions en termes de fonctionnalités et de performances lorsque l'interface est agréable à utiliser et qu'il ne perd pas de temps à apprendre à s'en servir. Posons-nous la question de savoir ce qui fait la force actuelle de Microsoft, l'un des plus grands fournisseurs de logiciels. A prix comparable et fonctionnalités comparables, un utilisateur ayant déjà d'autres logiciels de Microsoft, choisira le logiciel de Microsoft au lieu du concurrent puisqu'il sait que l'utilisation du logiciel de Microsoft ressemblera à ceux qu'il a déjà utilisés et qu'il prendra moins de temps à apprendre à l'utiliser. C'est aussi une raison pour laquelle certaines entreprises ne veulent pas changer de logiciel, le coût d'apprentissage pour un nouveau logiciel (même s'il est nettement meilleur) est trop élevé pour justifier la migration.

1.1.3 Application

Dans le monde de l'informatique et plus précisément celui des interfaces utilisateurs, l'ergonomie prend une place de plus en plus importante. Que ce soit au niveau du matériel informatique comme l'apparition de souris et du clavier de plus en plus ergonomique tel qu'illustré sur à la Figure 2 [Cor05], ou au niveau logiciel, au niveau des interfaces. Comme dit précédemment pour Microsoft, le « look and feel », c'est-à-dire la façon dont sont présentés les composants des interfaces et leur navigation doivent être constants d'un logiciel à l'autre, et d'autant plus entre les logiciels d'une même compagnie. Microsoft, IBM, Apple et d'autres producteurs de logiciels ont émis des « guidelines » concernant le « look and feel » de leurs logiciels [Mic95], [App87], [Ibm92]. Comme nous l'avons vu, l'ergonomie ou l'utilisabilité rendent les interfaces plus efficientes, plus efficaces. C'est dans cet objectif, que les auteurs des livres et sites web fournissent différentes règles d'ergonomie et conseils de conception.

Figure 2 - Clavier ergonomique

[2] Parfois c'est le contraire : les utilisateurs expérimentés préféreront Unix pour les performances, alors qu'un utilisateur débutant préférera Macintosh pour son utilisabilité élevée.

1.2 Accessibilité

Comme nous l'avons fait pour l'ergonomie, nous présentons un mauvais exemple d'accessibilité. Par exemple, la première page du site de l'UCL (http://www.ucl.ac.be) que nous avons repris à la Figure 3, avec, à côté, la visualisation de cette même page avec Lynx Viewer, un simulateur du navigateur Lynx en ligne [Del03]. Lynx est un navigateur textuel qui est utilisé par les personnes aveugles.

Nous avons indiqué quelques-unes des correspondances afin d'illustrer quelques erreurs. Les trois premières correspondances indiquées en vert et par les numéros 1, 2 et 3 sont de bons exemples. Les deux dernières, indiquées en rouge et par les numéros 4 et 5 sont des mauvais exemples.

- **Les bons exemples**

1. Les textes des menus animés sont biens transmis par le navigateur Lynx.

2. L'image principale contient un texte alternatif qui indique le texte « UCL – S'inscrire et se réinscrire ». Ce texte est bien visualisé par le navigateur lynx. Cette image répond donc bien à un des critères d'accessibilités qui est exprimé comme suit dans le WCAG 1.0 [WCAG99a].

   Provide a text equivalent for every non-text element

3. La troisième correspondance est celle entre l'image d'une ligne rouge et le texte [ligne-rouge.gif]. Notons que cette image ne vérifie pas la règle que nous venons d'énoncer. Mais, puisque c'est une image de décoration et qui ne convoie pas d'information importante, c'est acceptable. D'autant plus que le titre du fichier image explique bien l'image.

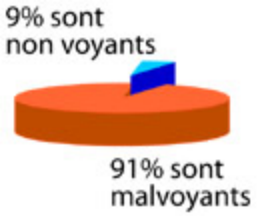
- **Les mauvais exemples**

4. Comme mauvais exemples, nous avons l'image de la ville de Louvain-la-Neuve qui ne contient ni de texte alternatif, ni un nom d'image significatif. Même si l'image n'apporte pas beaucoup d'information, cette image ne respecte pas la règle énoncée.

5. Pour ce qui est des informations encadrées, il s'agit en fait d'une image composée de plusieurs zones d'images cliquables. L'image principale contient un texte alternatif « outils ». Mais les informations, les liens contenus dans les zones d'images cliquables ne sont pas transmis dans le navigateur Lynx. Le site échoue donc dans l'évaluation de la règle suivante : *Provide redundant text links for each active region of a image map.*

Figure 3 - Comparaison entre la visualisation d'un site web par un navigateur classique et un navigateur textuel

On pourrait supposer évidement que rares sont les personnes qui utilisent ce type de navigateur, mais on se trompe. Par exemple, il y a plus de 6 millions de personnes handicapées en France dont 1,2 million de personnes déficientes visuelles [Kin05]. Parmi les déficients visuels, 9% sont totalement non voyants.



Par ailleurs selon une étude pour l'accès à l'emploi en Europe réalise en 2003, on a compté, dans les 15 pays de l'Union européenne, 37 millions de personnes ayant un handicap, soit **10% de la population.** [Hou03]

1.2.1 Définition

Nous avons regardé quelles sont les définitions que nous donne Google pour le terme anglais « Accessibility ».

1. Accessibility refers to ensuring that Content is accessible, i.e. ensuring that Content can be navigated and read by everyone, regardless of location, experience, or the type of computer technology used. Accessibility is most commonly discussed in relation to people with disabilities, because this group are most likely to be disadvantaged if the principles of accessible Web design are not implemented. Failure to follow these principles can make it difficult or impossible for people with disabilities to access Content. Creating accessible Content should be an integral part of the Web design philosophy, and accessibility features should be incorporated into all aspects of the design process. Testing for accessibility should also be incorporated into any and all user testing regimes, and should never be seen as an isolated event that can occur after other user testing has taken place. Designing for accessibility is thus as much a strategic issue as a purely technical one.
   www.murdoch.edu.au/cwisad/glossary.html

2. The degree to which software can be used comfortably by a wide variety of people, including those who require assistive technologies like screen magnifiers or voice recognition. […].
   docs.sun.com/db/doc/805-4368/6j450e60a

3. Ability to reach a destination or use a facility or service without being impeded by physical or other barriers due to auditory, visual, mobility, or cognitive disabilities.
   www.ctps.org/bostonmpo/mpo/gloss.htm

Ces trois définitions donnent un bon aperçu de l'étendue du terme « accessibilité ». Dans le cadre d'un site web ou d'une application informatique, nous pourrions définir l'accessibilité comme suit :

L'accessibilité d'une interface ou d'un site web désigne sa capacité à pouvoir être consulté par tous et dans toutes les conditions.

« Etre consulté par tous » signifie que tout utilisateur, victime d'un handicap ou non, doit pouvoir accéder à son contenu sans problème.

« Dans toutes les conditions » se rapporte à la configuration matérielle de l'utilisateur, qui ne doit pas être un obstacle à la navigation de l'interface. [Tho05]

L'accessibilité est, par sa définition, proche de l'ergonomie. S'adapter à la taille de l'écran, fournir le contenu en plusieurs langues, sont des règles d'ergonomie qui améliorent également l'accessibilité de l'interface. L'accessibilité fait en quelque sorte partie de l'ergonomie, mais s'en distingue aussi. En effet, on mesure l'utilisabilité d'une interface dans un contexte particulier, c'est-à-dire par un utilisateur particulier, sur une plate-forme particulière, dans un environnement particulier, alors que l'accessibilité se soucie de rendre le contenu accessible quel que soit le contexte.

1.2.2 Objectif

L'objectif de l'accessibilité est de permettre à tous de participer à la vie active, de faire en sorte que le handicap dont certaines personnes sont victimes ne les en exclut pas. L'informatique et Internet en particulier sont devenus des éléments essentiels à la vie active. Quel que soit le métier que l'on pratique, on est confronté à tout moment à l'informatique. Du magnétoscope au guichet de banque électronique, du guichet de train au commerce en ligne, nous sommes tous confrontés à des interfaces utilisateurs. Si l'accessibilité de l'informatique et d'Internet est si important, c'est parce qu'Internet est une ressource croissante d'information et une ressource de plus en plus importante dans beaucoup d'aspect de la vie, que ce soit dans le domaine de l'éducation, du commerce, de la politique gouvernementale, sociale, récréative et bien plus encore. Il est donc essentiel de donner les mêmes accès et les mêmes opportunités aux personnes victimes d'un handicap. L'autre raison de l'importance de l'accessibilité de l'informatique est qu'Internet, plus que tout autre média, est capable de s'adapter à l'utilisateur. C'est-à-dire qu'il existe des aides techniques, des outils, des logiciels qui permettent aux personnes handicapées de surmonter leurs handicaps et d'accéder aux contenus.

1.2.3 Application

Cet objectif se traduit par des recommandations, des règles pour la conception des interfaces utilisateurs. Ces règles sont émises par des organismes, des associations.

1.2.4 Aides techniques

Les personnes victimes d'un handicap peuvent, elles aussi, utiliser Internet mais elles ont besoin d'outils particuliers. Ces outils sont appelés des « aides techniques[3] ». Selon le handicap, l'utilisateur utilisera différentes aides techniques [WAI04].
D'autre part, outre ces aides techniques, les personnes victimes d'un handicap font souvent l'usage de navigateur textuel ou auditif. C'est-à-dire des navigateurs web qui transforment le site web en une version contenant uniquement du texte ou en des versions auditives qui peuvent être écoutées par l'utilisateur [WAI04].

[3] Assistive technologies in English.

1.3 Règles

Ce que nous appelons règles correspond à une recommandation ou un conseil plus ou moins concret issu de résultats expérimentaux, de prédictions issues de théories de l'activité humaine, de principe de psychologie cognitive, de sens commun, d'expérience pratique, etc. Les règles peuvent être très concrètes et objectives comme « Accompagner l'icône de son nom » [Nog02] par exemple, ou, au contraire très abstraites et subjectives : « Provide Useful Content[4] » [Koy03].

1.4 Validation automatique

Le meilleur outil pour évaluer l'utilisabilité d'un logiciel ou d'une interface est le test d'utilisabilité. Il s'agit de placer l'utilisateur final en situation effective d'utilisation du logiciel. Pour évaluer l'accessibilité d'une interface, on agit de manière similaire, c'est-à-dire que l'on place un utilisateur final victime d'un certain type de handicap en situation et on observe si l'utilisateur muni de l'aide technique adaptée éprouve encore de la difficulté à utiliser l'interface. Evidement, cela prend beaucoup de temps et d'argent. A partir des guidelines émis par les différentes sources, des méthodes d'évaluation ont vu le jour. Le site usabilis.com présente son mode opératoire (voir à la Figure 4) comme suit : « L'évaluation ergonomique consiste à passer en revue chacun des composants de l'interface. L'évaluateur vérifie si le composant respecte un ensemble de critères ergonomiques, appelé 'grille de critères' » [Nog05].

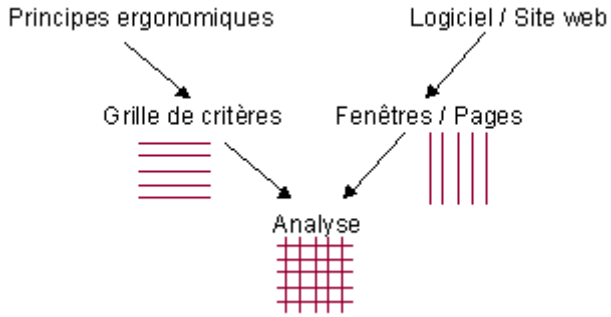


Figure 4 - L'évaluation ergonomique consiste à évaluer chacun des composants de l'interface vis-à-vis d'une grille de critères.

Mais des outils de validation **automatique** ont également vu le jour comme WebXACT [Wat04a], A-Prompt[Apr02], LIFT[Lif05] ou Sherlock[Mah97]. Au lieu de faire évaluer le logiciel par un véritable utilisateur, l'on vérifie que l'interface a certaines propriétés qui contribuent à l'utilisabilité et l'accessibilité du logiciel. La validation automatique consiste à parcourir la structure de l'interface de manière systématique et de vérifier les valeurs des attributs, de vérifier la structure des objets, etc. Pour cela il faut que les règles soient traduites en un langage qui permet la validation automatique.

[4] Fournir du contenu utile

1.4.1 Avantages

Les avantages de la validation automatique sont multiples [Ivo01], nous en présentons quelques-unes ci-dessous :

- une réduction importante des coûts car les méthodes automatiques font le travail plus rapidement.
- une meilleure uniformité des erreurs découvertes, ainsi qu'une prévision du temps et du coût des erreurs à travers une conception complète. Il n'est, en effet, pas simple de vérifier l'entièreté d'un logiciel par une méthode de validation non automatique.
- Une réduction du besoin d'expert en validation. Automatiser quelques aspects de validation telle que l'analyse ou les activités critiques permettent d'aider les concepteurs qui n'ont pas l'expertise dans ces aspects d'évaluation.
- La possibilité d'inclure l'évaluation à l'intérieur du processus de conception des interfaces utilisateurs, contrairement à l'application après implémentation. C'est important parce que l'évaluation avec la plupart des méthodes non automatiques peut typiquement être réalisée qu'après qu'un prototype ou l'interface aient été construits et tout changement à ce niveau sont bien plus coûteux [Nie93].

1.4.2 Inconvénients

Les inconvénients de la validation automatique viennent de sa limitation dans l'expression des règles principalement. En effet, certains aspects des interfaces graphiques ne peuvent pas être automatisés, tel que savoir si le texte d'un bouton sera compris ou non pas l'utilisateur. Christelle Farenc a montré que seulement 78% de l'ensemble des règles ergonomiques établies peut être évaluées par une validation automatique, et seulement 44% dans le pire des cas (voir à la Figure 5). [Far96]

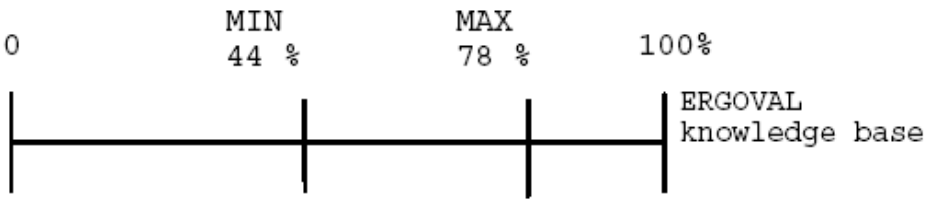


Figure 5 - Le pourcentage des règles ergonomiques évaluables automatiquement dans le meilleur et le pire des cas

Il faut bien se rendre compte que la validation automatique n'est sûrement pas aussi efficace que celle effectuée par un véritable utilisateur. Comme le dit très bien le site http://www.ergologique.com dans l'un de ses conseils : « il ne faut pas confondre ergonomie technologique et ergonomie par de vrais utilisateurs ». La conformité du code de l'interface par rapport aux recommandations, aux règles est une chose, mais elle ne permet pas à elle seule d'assurer l'ergonomie d'une interface utilisateur. « **La réflexion du concepteur est une étape primordiale pour l'application intelligente des guidelines concernant l'accessibilité et l'ergonomie**. » [Tho05]

1.5 En phase de conception

La plupart des outils de validation automatique que nous présenterons au chapitre 2 sont de type « post-design » [Gae05]. C'est-à-dire que ces outils de validation sont utilisés après la phase de conception de l'interface et se basent donc sur le rendu final de l'interface utilisateur. Par opposition, ce mémoire consiste en une analyse sur : « comment serait-il possible d'intégrer un outil de validation automatique de règles d'ergonomie et/ou d'accessibilité au cours de la phase de conception ou du moins à un niveau indépendant du rendu final de l'interface utilisateur ? ».

1.6 Méthodologie

La méthodologie choisie pour cette analyse, se structure en 4 étapes (voir Figure 6). Dans un premier temps nous allons balayer nos différentes sources pour former un ensemble de règles d'ergonomie et d'accessibilité. Cet ensemble sera aussi large, aussi général que possible. Cette recherche sera expliquée au chapitre 3. La seconde étape consistera à classer ces règles selon certains critères choisis. Ce sera le sujet du chapitre 4. Ensuite, nous tenterons de traduire, d'interpréter les règles classifiées en des règles adaptées pour un langage particulier. C'est la troisième étape de notre méthodologie. Ce sera développé dans le chapitre 5. Finalement, notre étude se penchera sur les possibilités d'implémentation d'un outil de validation automatique pour ce langage particulier. C'est notre dernière étape qui sera présentée dans le chapitre 6.

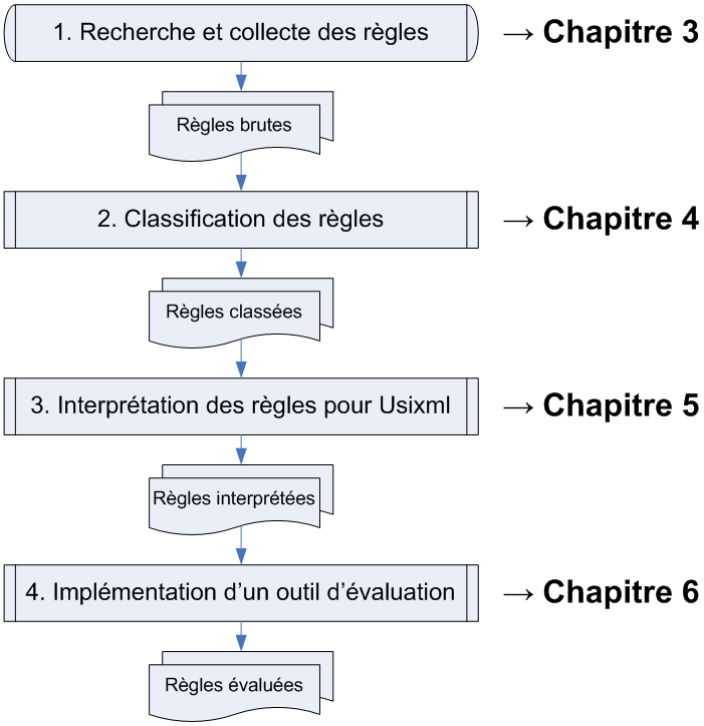


Figure 6 - Les quatre étapes et leurs résultats intermédiaires de notre méthodologie

1.6.1 Du général au spécifique

Au cours de notre démarche, les règles seront de plus en plus spécifiques. Le choix des critères de classification, le choix du langage, le choix des techniques d'implémentation vont faire que nos règles vont devenir de plus en plus précises, d'autant plus précises que le contexte d'application se définira. De même, au fur et à mesure que nous avancerons dans la démarche, nos règles générales se transformeront en plusieurs règles spécifiques.
En effet, les règles que nous récolterons lors de la première étape seront très généralement abstraites et subjectives, lors de la classification et de l'interprétation, ces règles seront affinées en des règles concrètes, objectives et spécifiques au langage choisi.

1.6.2 Une restriction inévitable

Au cours de notre démarche, nous serons obligés de restreindre l'ensemble de nos règles. D'une part parce qu'il y en a des milliers et qu'il est impossible de les présenter toutes dans ce mémoire. D'autre part, parce que nous faisons des choix et « Choisir, c'est renoncer ». En effet, lors de la classification, certaines règles seront mises de coté, pour plusieurs raisons telles que : « pas classifiable », « pas pertinente pour cette analyse », « pas applicable aux interfaces utilisateurs », etc. De même, le choix d'un langage pour l'interprétation des règles va restreindre notre ensemble de règles. Aucun langage ne permet d'exprimer toutes les interfaces utilisateurs possibles à lui seul. Un langage est limité par sa structure, son étendue d'application, etc. C'est encore une explication à la restriction inévitable de notre ensemble de règles. Finalement, l'implémentation d'un outil de validation automatique, sera restreinte par les techniques utilisées pour exécuter la validation comme nous le verrons au chapitre 6.

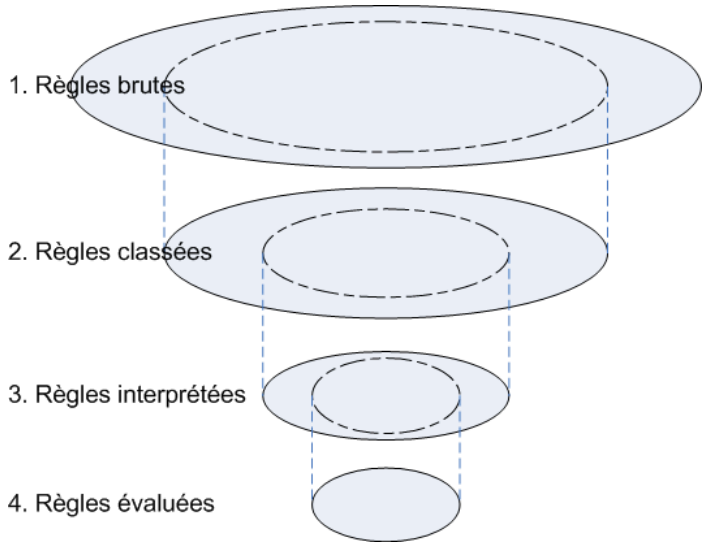


Figure 7 - Chaque étape de la méthodologie restreint inévitablement l'ensemble des règles.

Chapitre 2 Etat de l'art

Dans l'introduction, nous avons défini notre méthodologie de travail. A partir de cette méthodologie en 4 étapes, il est intéressant d'observer ce qui a déjà été effectué. C'est l'objet de ce chapitre. Il est évident que d'autres études ont déjà été entreprises pour collecter, classifier, interpréter et évaluer les règles d'ergonomie et d'accessibilité. Parfois elles se sont concentrées sur une étape de la méthodologie, parfois sur plusieurs d'entres elles.

Pour ce chapitre, nous avons décidé de nous concentrer sur quelques outils de validation automatique.

Dans la section 2.1, nous présenterons un premier outil, WebXACT anciennement Bobby. La section 2.2, aura pour sujet A-Prompt, un autre outil de validation de règles d'accessibilité. Dans la section 2.3, d'autres études ou outils seront brièvement esquissés.

2.1 WebXACT

WebXACT est plus connu sous le nom de Bobby. Il s'agit de l'outil de vérification d'accessibilité des sites web le plus connu. Sa première version date de septembre 1996. A ce moment il se nommait encore Bobby et était un produit gratuit offert par CAST, The Center for Applied Special Technology [Cas05]. En 2001, il passe en version 3.3 et devient payant. En septembre 2002, Watchfire Corporation, un fournisseur en service et logiciel de gestion achète Bobby à CAST. CAST étant une organisation sans but lucratif, a estimé que Bobby sera plus utile et servira mieux les réalisateurs de site web si l'outil était confié à une compagnie qui est spécialisée dans le développement de logiciel et qui possède une bonne expérience dans le marketing de produit. Le produit WebXACT existe dans une version hors ligne très complète mais payante, et une version gratuite disponible en ligne (voir Figure 8).

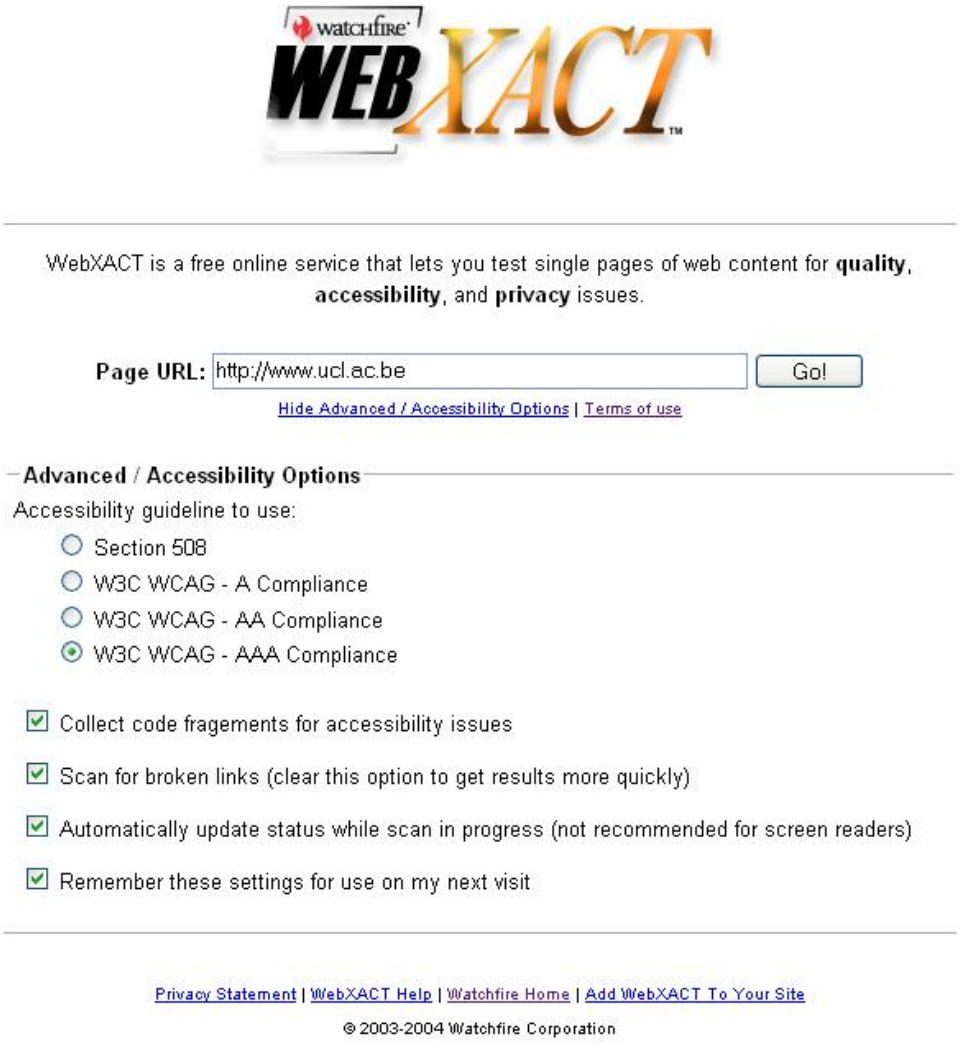


Figure 8 - WebXACT dans sa version en ligne

- **Collation**

WebXACT se base sur les deux grandes sources de directives concernant l'accessibilité des sites web, à savoir la section 508[508] et les directives de la WAI (Web Accessibility Initiative) [WAI98]. Nous parlerons de ces deux sources dans le chapitre 3.

- **Classification**

Cependant, la vérification qu'il propose peut se faire soit pour les directives de la section 508, soit pour les directives WCAG 1.0 du WAI, avec la possibilité de choisir une compatibilité avec les 3 niveaux de priorité des directives WCAG 1.0. Il s'agit donc de 3 groupes, les directives de priorité 1 pour obtenir la WAI Conformance Level "A", les directives de priorité 1 et 2 pour obtenir la WAI Conformance Level "Double-A" et finalement toutes les directives, de priorité 1, 2 et 3 pour obtenir la WAI Conformance Level "Triple-A"

- **Interprétation**

Puisque les règles d'accessibilité de la section 508 et celles contenues dans le WCAG 1.0 sont déjà assez fort dirigées vers le langage HTML, les concepteurs du logiciel WebXACT n'ont pas beaucoup de difficulté à interpréter les directives. En effet, le WAI fourni également des techniques HTML, CSS, etc. [W3C00a] [W3C00b] [W3C00c] pour s'assurer que le document que l'on évalue vérifie les directives voulues. WebXACT a cependant réorganisé et développé les directives du WCAG et de la section 508 en 94 points de contrôles. [Wat04b]

- **Implémentation**

WebXACT est écrit en Java pour la version off line et en Java script pour la version en ligne. D'ailleurs un utilisateur a remarqué que le site de vérification en ligne de WebXACT utilise donc Java script pour fonctionner correctement. Mais justement, une des recommandations du WCAG 1.0, la directives numéro 6.3 de priorité 1 *("Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported*") demande de fournir une alternative au Java script, puisque certains groupes de personnes handicapées désactivent cette technologie. [Pil03]

- **Résultat**

Les résultats que renvoient WebXACT doivent être interprétés avec un soin particulier (voir Figure 9). En effet, WebXACT attire l'attention sur beaucoup de problèmes potentiels, c'est-à-dire des règles que le logiciel ne permet pas d'évaluer. Nous le verrons plus tard, toutes les règles ne sont pas automatisables. Malheureusement, on ne peut, en général, pas dire si on a réellement une erreur ou non pour ces règles. Dans beaucoup de cas, il n'est pas nécessaire de faire attention aux éléments que propose WebXACT, mais bien plutôt de vérifier les facteurs de risques et de s'assurer qu'on les a manipulés correctement.

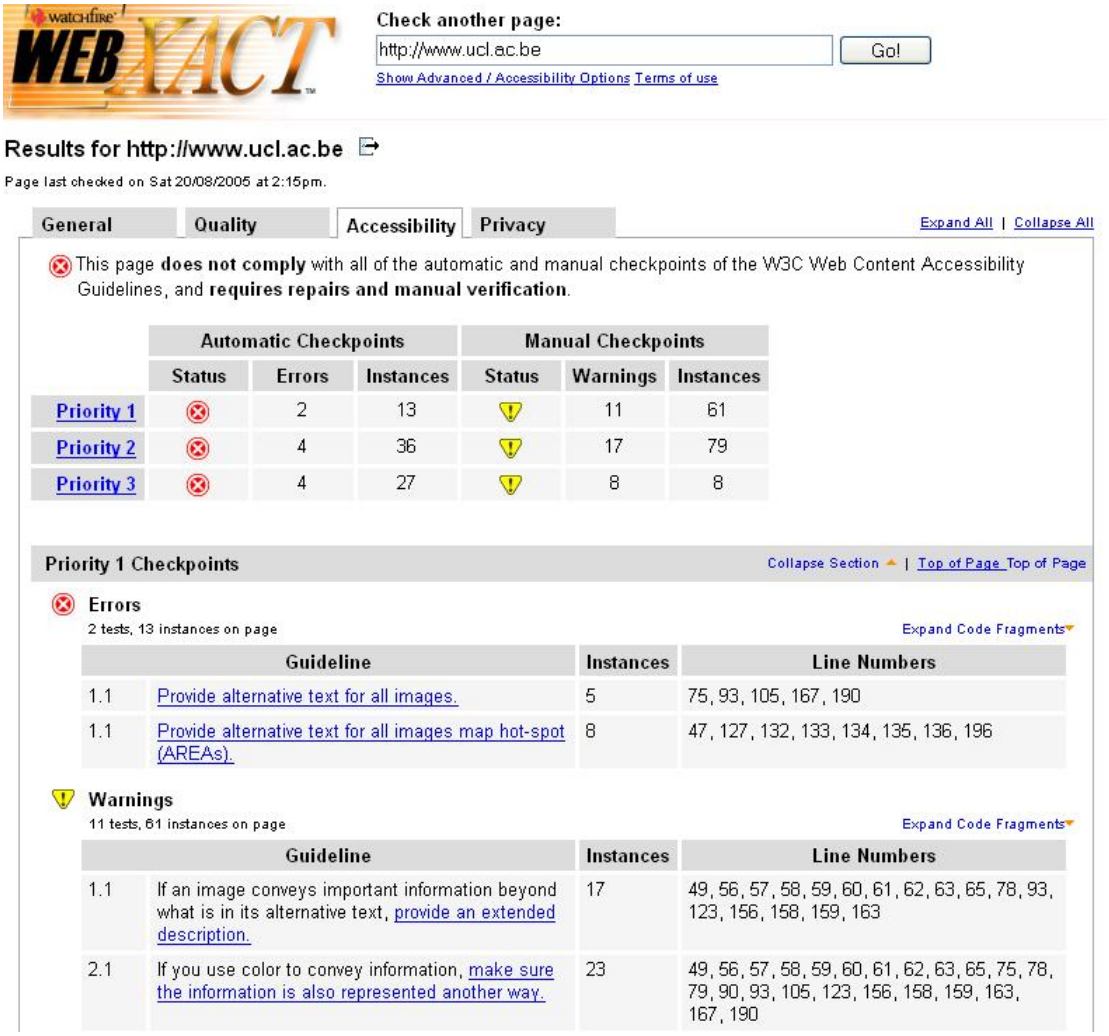


Figure 9 - Les résultats donnés par WebXACT dans sa version en ligne

WebXACT ne fait que détecter les erreurs et ne propose pas beaucoup d'aide pour les réparer. Le prochain outil que nous présentons permet la correction des erreurs signalées.

2.2 A-Prompt

A-Prompt : Web Accessibility Verifier [Apr02] a été développé pour aider des concepteurs de site web à améliorer l'accessibilité et la rentabilité des documents HTML. En examinant les barrières à l'accessibilité des pages des sites web et en proposant des corrections aux problèmes, la volonté d'A-Prompt est de toucher le plus de personnes possible afin de s'assurer qu'une plus grande majorité des sites web soient accessibles. Ce logiciel est rendu disponible par une collaboration commune entre le centre adaptatif de ressource de technologie (ATRC) à l'université de Toronto et le centre de TRACE à l'université du Wisconsin.

- **Collation**

Tout comme WebXACT, A-Prompt se base sur les directives WCAG 1.0 du WAI et sur les directives de la section 508 également.

- **Classification**

La classification effectuée par A-Prompt est similaire à celle effectuée par WebXACT (voir Figure 10).

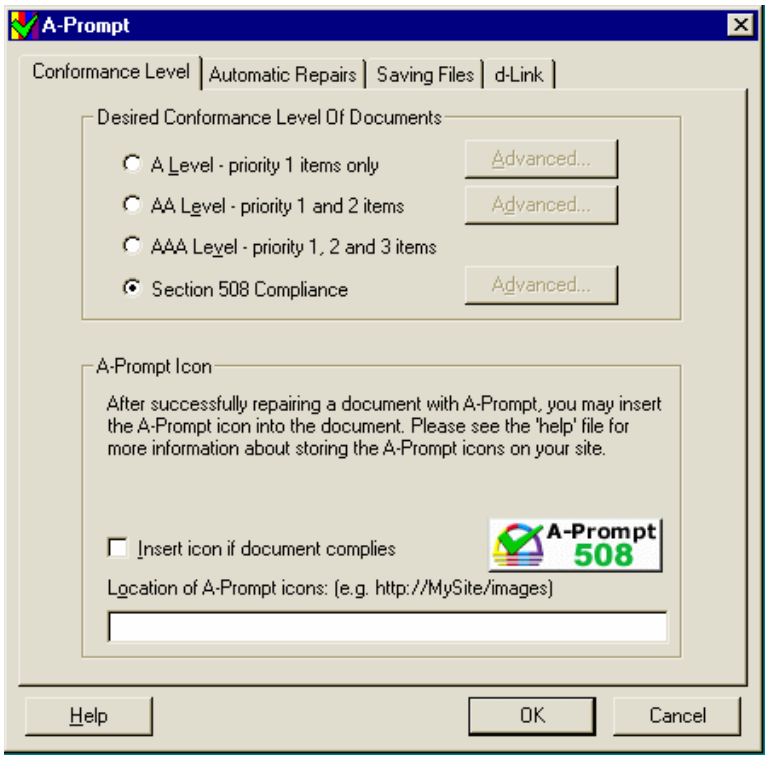


Figure 10 - A-Prompt permet de choisir entre les niveaux de priorité du WAI ou la section 508

- **Interprétation**

Concernant l'interprétation, A-Prompt a réorganisé et interprété les directives différemment et tient compte dans sa version actuelle de 56 réparations. Ces réparations sont par exemple : si « Alternate text may be too long » alors une fenêtre permettant de modifier/réparer le texte alternatif est proposée à l'utilisateur.

- **Implémentation**

L'implémentation d'A-Prompt est réalisée en Java. Nous n'avons pas beaucoup d'information concernant les techniques utilisées par A-Prompt pour évaluer le code HTML.

- **Résultat**

L'avantage d'A-Prompt est de pouvoir corriger le document HTML automatiquement ou avec assistance de l'utilisateur afin de le rendre compatible aux directives voulues.

Prenons par exemple, la directive du WAI, qui exige que l'on fournisse un titre à chaque cadre d'un document. L'interprétation que l'on fait de cette directive pour le langage HTML est que chaque balise frame doit avoir un attribut title. Une version correcte serait l'exemple suivant :

<frame src="ucl_toc.html" title="Table of Contents frame">

Si l'attribut title n'est pas inclus, A-Prompt présentera une fenêtre de dialogue comme présenté à la Figure 11, fournissant les informations concernant le fichier source de la frame et le nom si ce dernier est disponible.

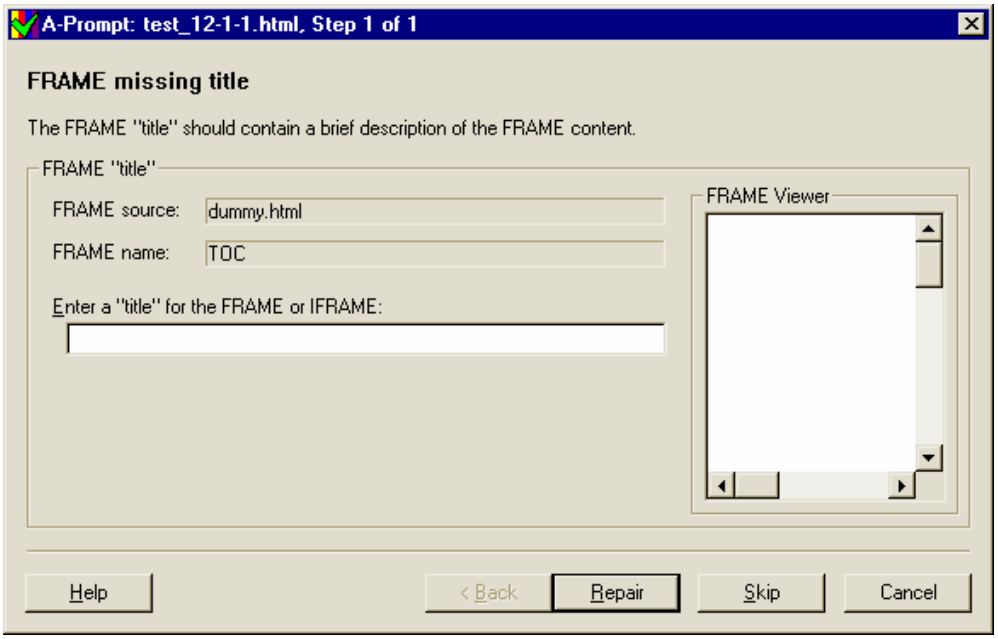


Figure 11 - Fenêtre de dialogue du logiciel A-Prompt

Nous pouvons voir que la fenêtre de dialogue inclut un champ d'édition que l'utilisateur superviseur peut remplir avec un contenu adéquat. De cette manière, l'utilisateur corrige, à la volée, les erreurs découvertes par le logiciel. En effet, A-Prompt ajoutera automatiquement la description entrée par l'utilisateur dans le code source du document HTML en insérant un attribut title dans le frame corrigée.

2.3 Autres outils ou études

D'autres outils et études correspondant à chacune des étapes de notre méthodologie ont également été réalisés.

- **Collation**

Au niveau de la collation des règles, on peut citer le WAI et la section 508 qui sont fortement utilisés par les outils de validations automatiques et qui proposent principalement des règles d'accessibilité. Mais également Microsoft, Apple, SUN qui proposent des règles d'ergonomie pour l'ensemble de leur logiciel et de leur « look and feel ». On peut également citer les études de collation de règles ergonomiques générales tel que Sidney L. Smith et Jane N. Mosier qui ont écrit un recueil de 944 règles en 1986 [Smi86], ou tout simplement le service de recherche technique de la Poste française, avec son guide de conception des interfaces graphiques édité en septembre 1994.

De plus, certaines études vont plus loin encore que des critères tels que ceux de WAI, en joignant l'utilisabilité à l'accessibilité, en définissant des critères d'utilisabilité de l'accessibilité [Lep02].

- **Classification**

Généralement, les organismes qui ont effectué une collation de règles se sont basés sur des critères pour les classifier. Parmi les classifications les plus courantes, nous avons :

- Classification selon les objets sur lesquels s'appliquent les règles, c'est-à-dire des éléments de l'interface, tels que boutons, radio boutons, cadres, texte d'entrée, contrôleur, etc.… Dans ce type de classification, nous pouvons signaler les 944 règles de Smith et Mosier [Smi86] ou les critères d'AccessiWeb [Acc03]
- Classification selon le processus de création, c'est-à-dire des règles qui s'appliquent à chaque étape du processus, du prototype aux choix des couleurs. Cette classification est reprise par le livre de Galitz [Gal96], de Jakob Nielsen [Nie93]
- Classification par critères ergonomiques. Plusieurs études ont tenté de réaliser une classification des règles d'ergonomie selon des critères ergonomiques. On peut parler de Bastien et Scapin qui ont défini 8 critères généraux d'ergonomie que nous présenterons au chapitre 4.

- **Interprétation**

L'interprétation qui consiste à traduire les principes, les directives pour une technologie particulière ont également fait l'objet d'étude. L'exemple le plus évident est l'interprétation que font AccessiWeb et BlindSurfer des recommandations du WAI pour le langage HTML.
Concernant d'autres langages que le web, nous pouvons signaler le travail d'IBM pour le langage Java [Ibm00], celui de Microsoft avec son Microsoft Active Accessibility [Mic05] qui est destiné, entre autres, aux langages C et C++.
On se rend compte que l'interprétation des règles pour un langage particulier sous-entend souvent la possibilité de tester automatiquement les règles interprétées. Ce que nous voulons dire, c'est qu'il est intéressant d'effectuer une interprétation pour un langage particulier dans la mesure où ce travail peut servir. La validation automatique qui apporte

son lot d'avantages est une raison suffisante pour lancer une étude sur l'interprétation des règles. Il n'est donc pas étonnant de voir que la plupart des interprétations le sont pour des langages de types « XML » puisque leur structure permet de les consulter de manière plus aisée. Nous ne serons pas surpris de voir dans les prochaines années des études similaires pour des langages comme XUL [Hya01], XAML (Microsoft Extensible Application Markup Language) ou XIML [Pue02] pour ne parler que de ces trois là.

- **Implémentation**

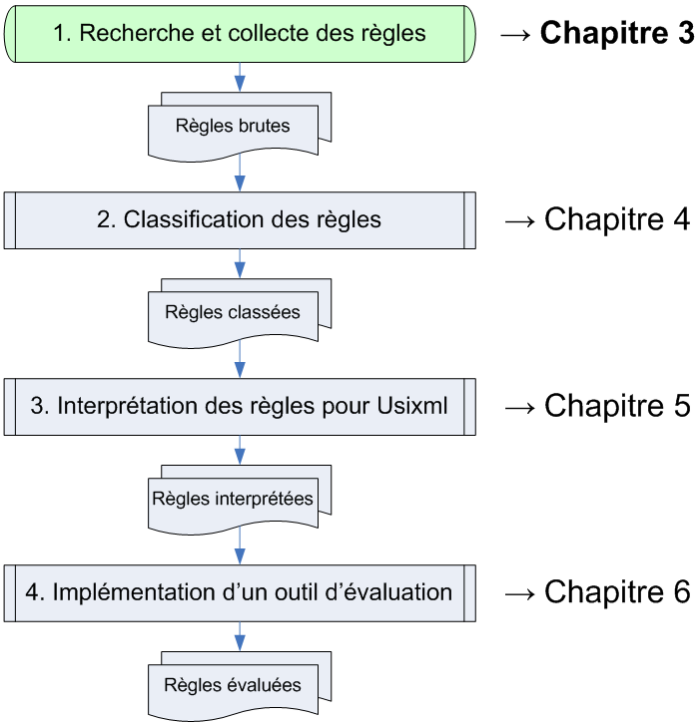
Concernant l'implémentation, nous n'avons pas observé beaucoup d'études et de réalisations, à l'exception des outils de validation des pages web, qui sont nombreux. Une des raisons possibles est que, comme nous l'avons dit au point précédent, le langage utilisé pour les sites web est du type XML, c'est-à-dire un langage à balise qui est particulièrement facile à traiter.

Cela dit, même pour le langage XML, il existe beaucoup de techniques permettant une implémentation d'un outil d'évaluation tel que le montre l'étude de Vicente Luque Centeno [Cen05] dont nous nous inspirons fortement au chapitre 6.

Pratiquement tous les langages sont possibles à utiliser pour une implémentation d'un outil de validation pour des documents XML, prenons par exemple Tidy [Rag03] qui est écrit en C, WebXACT, A-Prompt ou Wave[Kas05] qui sont écrits en Java ou encore Webxref qui est écrit en Perl [Jan95]

Chapitre 3 Recherche et collecte des règles

La recherche et la collecte des règles constituent la première étape de notre méthodologie. Lors de notre recherche, nous nous sommes intéressés aux organismes, personnes et associations qui ont déjà réalisé un travail d'analyse sur les règles d'ergonomie et d'accessibilité. Ces différentes sources seront présentées dans les sections suivantes. Puisque nous nous concentrons sur l'accessibilité, nous présenterons à chaque fois la section concernant l'accessibilité avant celle traitant de l'ergonomie. Dans la section 3.1, nous expliquerons le problème de l'exhaustivité et du pourquoi ce n'est pas possible de considéré toute les règles. Dans la section 3.2 nous traiterons des sources contenant des règles d'accessibilité. Nous finirons avec la section 3.3 où nous traiterons des sources contenant des règles d'ergonomie. Lorsque nous aurons compulsé les différentes sources, nous serons à même d'effectuer une classification de ces règles. Ce dernier point fait l'objet du chapitre suivant.



3.1 Introduction

Il est impossible de présenter dans cette étude toutes les règles d'ergonomie et d'accessibilité qui ont été émises, d'une part parce que la quantité est énorme et diversifiée et d'autre part, parce qu'il y a une redondance importante entre les différentes sources de ces règles. Cette redondance est liée au fait que chacune des sources, dont certaines que nous présenterons, tente de couvrir le sujet à sa manière, avec ses termes et de la manière la plus complète possible.

Puisque les sources sont nombreuses, nous avons déjà fait une présélection. En effet, nous avons décidé de nous concentrer principalement sur les règles d'accessibilité. Ces règles constituent un ensemble plus restreint, plus facile à couvrir dans le cadre de cette analyse. Comme nous l'avons vu dans le premier chapitre, l'accessibilité est parfois considérée comme un cas particulier de l'ergonomie et parfois comme une extension de l'ergonomie. A vrai dire, elle est les deux. Beaucoup de règles d'ergonomie sont des règles d'accessibilité. Une interface hautement utilisable par l'application de règles d'ergonomie, rend l'accès à l'information plus aisée qui est le but même des règles d'accessibilité. Comme nous pouvons le voir à la Figure 12, l'intersection des deux ensembles de règles est très importante. Seule une petite partie des règles d'accessibilité est propre à l'accessibilité et ne constitue pas un ensemble de règles d'ergonomie.

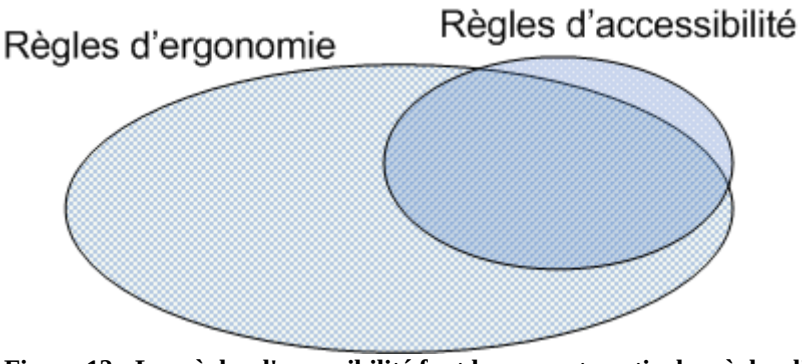


Figure 12 - Les règles d'accessibilité font largement partie des règles d'ergonomie

3.2 Les sources des règles d'accessibilité

De plus en plus d'organismes émettant des règles d'accessibilité ont vu le jour avec l'avènement d'Internet. Comme nous l'avons souligné au premier chapitre, Internet est une source grandissante d'information et prend une part importante dans beaucoup de domaine comme l'éducation, le commerce, le loisir, etc. C'est devenu un outil d'information et de communication indispensable.
Des associations tel que BlindSurfer [Bli05] ou BrailleNet [BrN05] tentent de conscientiser les instances gouvernementales et les concepteurs de sites web afin de développer des sites web accessibles aux personnes victimes d'un handicap. D'ailleurs plusieurs pays ont inscrit l'accessibilité dans leur législation, comme les Etats-Unis avec

la loi « Americans with Disabilities Act » [Ada90] ou la France avec la circulaire « relative aux sites Internet des services et des établissements publics de l'Etat ».
Afin d'aider les concepteurs de sites web à rendre leurs sites web accessibles, en accord avec la loi, des règles et des techniques ont été écrites par diverses associations et organisations.

3.2.1 Web Accessibility Initiative (WAI)

Sans doute l'organisme le plus connu concernant l'accessibilité des sites web. Le WAI fut créée en février 1997 par le World Wide Web Consortium (W3C) [Jac05], et regroupe aujourd'hui plus de 500 membres. Les membres de WAI sont des organisations issues de l'industrie, des organismes pour personnes handicapées, de la recherche et de gouvernements.

Le but poursuivi par cet organisme peut s'expliquer par une citation de Tim Berners-Lee, le directeur du W3C et inventeur du World Wide Web.

The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.

Afin d'accomplir ce but, le WAI développe un ensemble de recommandations ainsi que des outils, des technologies pour aider à comprendre et implémenter l'accessibilité de sites web. Nous nous intéressons principalement aux recommandations émises par cet organisme. Le but des directives du WCAG est d'expliquer comment rendre le contenu des sites web accessible aux personnes victimes d'un handicap. Le contenu des sites web fait généralement référence aux informations contenues dans les pages web et les applications web. Ceci incluant les textes, les images, les formulaires, les sons, etc.

3.2.1.1 WCAG 1.0

- **Titre**

Les initiales **WCAG** signifient **W**eb **C**ontent **A**ccessibility **G**uidelines. Le numéro 1.0 signifie qu'il s'agit de la première version. Cette première version du document traitant des recommandations fut approuvée en mai 1999 et constitue la version stable de référence actuelle.

- **Provenance**

Cet ensemble de règles est émis par le WAI (Web Accessibility Initiative).

- **Statut**

Le statut de ce document est relativement important. En effet, il est très largement reconnu et adopté par la communauté internationale comme un standard pour l'accessibilité des sites web.

- **Nombre de règles**

Cet ensemble contient 65 points de contrôle, qui sont structurés en 14 principes généraux de conception de l'accessibilité. Chacun des points de contrôle explique comment appliquer le principe général dont ils font partie [WCAG99b]. A chaque point de contrôle

est assignée une priorité de 1 à 3. [WCAG99c] La signification des priorités est la suivante :

- **Priorité 1** : Un développeur de contenu Web **doit** satisfaire ce point de contrôle. Sinon, pour un ou plusieurs groupes, il sera impossible d'accéder à l'information du document. Satisfaire ce point de contrôle est une exigence élémentaire.
- **Priorité 2** : Un développeur de contenu Web **devrait** satisfaire ce point de contrôle. Sinon, un ou plusieurs groupes auront des difficultés d'accès à l'information du document. Satisfaire ce point de contrôle lèvera certaines barrières empêchant l'accès des documents Web.
- **Priorité 3** : Un développeur de contenu Web **peut** respecter ce point de contrôle. Sinon un ou plusieurs groupes auront quelques difficultés pour accéder à l'information du document. Satisfaire ce point de contrôle améliorera l'accès aux documents Web.

De plus, pour chaque point de contrôle, des techniques d'implémentation en vue de satisfaire le point de contrôle sont indiquées.

- **Exemples**

> ***Guideline 2. Don't rely on color alone.***
> *Ensure that text and graphics are understandable when viewed without color.*
> ***Checkpoints:***
> *2.1 Ensure that all information conveyed with color is also available without color, for example from context or markup.*
> *2.2 Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen.*

- **Avantages/ Inconvénients**

L'avantage de cette source est qu'elle est largement documentée et utilisée par les différents outils d'évaluation d'accessibilité des sites web.

3.2.1.2 WCAG 2.0

- **Titre**

Le titre WCAG 2.0 [WCAG05] signifie la même chose que la source précédente. Le numéro 2.0 signifie uniquement qu'il s'agit de la seconde version. Cette seconde version du document traitant des directives du WAI, est encore en cours de développement et devrait être complétée pour 2005.

- **Provenance**

Cet ensemble de règles sont émises par le WAI (Web Accessibility Initiative). Plus exactement par le Web Content Accessibility Guidelines Working Group (WCAG WG) du WAI.

- **Statut**

Le statut de ce document n'est pas encore réellement établi puisqu'il est toujours en cours de développement. Mais il est destiné à remplacer la première version et à prendre de plus en plus d'importance sur la scène internationale.

- **Nombre de règles**

La version 2.0 est organisée autour de 4 principes de conception pour l'accessibilité des contenus web.

1. Perceptible

S'assurer que le contenu peut être perçu par tout utilisateur. Par exemple, une image sans équivalent textuel ne peut être perçue par une personne aveugle. De même, un fichier sonore sans transcription textuelle ne peut être perçu par une personne sourde.

2. Utilisable

S'assurer que les éléments d'interface du contenu sont utilisables par tout utilisateur. Par exemple, une personne incapable d'utiliser une souris doit être en mesure de se déplacer dans le contenu. De même, un tableau de données ou un contenu non structuré sont difficilement navigables par une personne qui n'a pas une vision globale de l'écran et qui doit l'explorer pas à pas.

3. Compréhensible

Rendre le contenu et les contrôles compréhensibles par autant d'utilisateurs que possible. Par exemple, indiquer les changements de langue permet aux utilisateurs de synthèse vocale d'entendre prononcer le contenu dans la bonne langue. De même, un langage simple et des mécanismes de navigation cohérents rendent le contenu plus compréhensible pour les personnes ayant une limitation cognitive.

4. Robuste

Utiliser des technologies Web qui maximisent la capacité du contenu à travailler avec les technologies d'adaptation et les navigateurs actuels et futurs.

En dessous de chaque principe, des recommandations définissent comment appliquer le principe à une zone spécifique. Et pour chaque recommandation, sont regroupés des critères de succès, des définitions, les bénéficiaires et des exemples.

- **Exemples**

Principle 1: Content must be perceivable.

Guideline 1.1 Provide text alternatives for all non-text content.

Level 1 Success Criteria for Guideline 1.1

1. *For all non-text content that is used to convey information, text alternatives identify the non-text content and convey the same information. For multimedia, provide a text-alternative that identifies the multimedia.*
2. *For functional non-text content, text alternatives serve the same purpose as the non-text content. If text alternatives can not serve the same purpose as the functional non-text content, text alternatives identify the purpose of the functional non-text content*

3. *For non-text content that is intended to create a specific sensory experience, text alternatives at least identify the non-text content with a descriptive label.*
4. *Non-text content that is not functional, is not used to convey information, and does not create a specific sensory experience is implemented such that it can be ignored by assistive technology.*
5. *For live audio-only or live video-only content, text alternatives at least identify the purpose of the content with a descriptive label.*

Level 2 Success Criteria for Guideline 1.1

1. *No level 2 success criteria for this guideline.*

Level 3 Success Criteria for Guideline 1.1

1. *For prerecorded multimedia content, a combined transcript of captions and audio descriptions of video is available.*

- **Avantages/ Inconvénients**

L'avantage de cette version par rapport à la première version est qu'elle intègre d'autres buts que celui de promouvoir l'accessibilité aux contenus des sites web. Parmi ces buts, il y a celui de s'assurer que les recommandations émises soient applicables à travers les technologies (CSS, SMIL, SVG, MathML, XML, etc.) autres que l'HTML, mais également celui d'identifier clairement les bénéficiaires de l'application de chacune des recommandations, afin de montrer la correspondance la plus précise entre les besoins des personnes victimes d'un handicap et chaque recommandation. Par contre, dans un souci d'indépendance à la technologie, les règles sont devenues beaucoup plus générales et présentent une classification plus générale aussi. Cela rend malheureusement l'interprétation que l'on a d'elles beaucoup plus large au risque d'être parfois contradictoire.

3.2.2 La Section 508

L'autre source de règles qui est souvent utilisée par les outils d'évaluation des sites web est celle de la section 508 du gouvernement américain [508]. La sous partie B du standard de la section 508 concerne la partie technique et se divise en plusieurs domaines :

1. Les applications logiciels et les systèmes d'exploitation.
2. Les Intranet basés web et les systèmes et information d'Internet.
3. Les produits de télécommunication.
4. Les produits vidéo et multimédia.
5. Les produits tel que les kiosques d'information, les fax, les calculateurs …
6. Les ordinateurs portables et les ordinateurs de bureau.

- **Titre**

C'est uniquement aux deux premiers domaines que nous nous intéresserons. Il s'agit des directives pour les logiciels « **§ 1194.21 Software applications and operating systems.** » et celle pour les sites web « **§ 1194.22 Web-based intranet and internet information and applications.** »

- **Provenance**

Déjà en 1990, les Américains ont adopté une loi intitulée « Americans with Disabilities Act » qui interdit la discrimination et veut assurer une chance égale aux personnes ayant des incapacités en matière d'emploi ainsi que dans l'accès aux services gouvernementaux, aux équipements publics, aux installations commerciales et au transport [Ada90]. Avec l'avènement de l'informatique grand public et Internet, c'est en 1998 que le gouvernement fédéral américain publie la Section 508 du Rehabilitation Act qui exige que tout l'appareil gouvernemental adapte ordinateur, logiciels, produits multimédia et, bien sûr, les sites Web conformément aux règles de la section 508.

- **Statut**

C'est une loi. Cette liste de recommandation est stricte et exigeante, puisque toutes les institutions gouvernementales américaines doivent satisfaire à ces recommandations.

- **Nombre de règles**

La partie du standard qui s'occupe des logiciels et qui est intitulée « **§ 1194.21 Software applications and operating systems.** » contient 12 règles de (a) à (l). L'autre partie du standard qui s'intéresse aux sites web et intitulée « **§ 1194.22 Web-based intranet and internet information and applications.** » contient 16 règles de (a) à (p).

- **Exemples**

> ***§ 1194.21 Software applications and operating systems.***
>
> *(a) When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.*
>
> ***§ 1194.22 Web-based intranet and internet information and applications.***
>
> *(a) A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).*

- **Avantages/ Inconvénients**

Les règles de la section 508 sont strictes par leur caractères officiels, mais ont l'avantage (ou l'inconvénient, c'est selon) de pouvoir être interprétées de manière très large. Cet ensemble de règles est celui le plus utilisé en Amérique. D'ailleurs, la section 508 (1194.22) est souvent considérée comme la version américaine des règles de la WAI.

3.2.3 IBM

L'accessibilité est un point d'attention important pour IBM, que ce soit dans les produits qu'ils ont développés (Talking Typewriter de 1960, The 1403 Braille Printer de 1975), que ce soit dans le personnel qu'ils emploient (premier employé handicapé en 1920) ou dans la création de l'Accessibility Center, un centre mondial pour accélérer l'attention sur la conformité des produits [Ibm05a]. IBM a aussi émis certaines recommandations d'accessibilité à destination des développeurs, qui sont très proches de celles de la section 508[Ibm05b].

- **Titre**

IBM a émis des listes de recommandations pour trois types de technologie, celle des logiciels en général, des applications web et des applications Java. Les titres sont les suivants :

- IBM Software Accessibility Checklist - Version 3.5.1
- IBM Web accessibility checklist - version 3.5
- IBM Java accessibility checklist - version 3.1

- **Provenance**

IBM est une société qui a toujours été très proche du gouvernement américain et a fortement supporté l'usage par le gouvernement fédéral de l'accessibilité comme critère dans la procuration des technologies d'information électronique.

- **Statut**

En comparaison de la section 508 qui est une loi, les check-lists d'IBM qui sont fortement inspirés de la section 508 n'ont rien d'officiels et constituent plus une aide, une forte requête vis-à-vis des développeurs, pour qu'ils produisent une interface convenable au sens ergonomique du terme.

- **Nombre de règles**

La check-list pour les logiciels contient **18 recommandations** qui sont classées dans les catégories : « Keyboard access », « Object information », « Sounds and multimedia », « Display », « Timing » et « Verify accessibility ».

La check-list pour les applications web contient **16 recommandations**.

La check-list pour les applications java contient **19 recommandations** qui sont classées dans les catégories suivantes : « Keyboard access », « Object information », « Sounds and multimedia », « Display », « Timing », « Documentation » et « Verify accessibility ».

- **Exemples**

IBM Software Accessibility Checklist - Version 3.5.1

1. Keyboard access

1.1. Provide keyboard equivalents for all actions.

1.2. Do not interfere with keyboard accessibility features built into the operating system.

IBM Web accessibility checklist - version 3.5

12. Color & contrast. Ensure that all information conveyed with color is also conveyed in the absence of color.

IBM Java accessibility checklist - version 3.1

5. Timing.

5.1. Provide an option to adjust timed responses or allow the instruction to persist.
5.2. Avoid the use of blinking text, objects, or other elements.

- **Avantages/ Inconvénients**

L'intérêt que nous trouvons à prendre en compte les recommandations d'IBM est que ce dernier a exprimé les recommandations pour le web (très similaires à celle de la section 508 pour les applications web) mais aussi pour les logiciels en général et pour les logiciels utilisant Java™. Cela nous permet d'observer une certaine abstraction par rapport au langage utilisé, puisque le WCAG 1.0 s'applique principalement au langage HTML.

3.2.4 BlindSurfer

- **Titre**

Créé dans le cadre d'un projet d'accessibilité du web, **BlindSurfer** est un label d'accessibilité [Bli05].

- **Provenance**

C'est en 1999 que l'association BlindSurfer est créée à l'initiative d'un utilisateur aveugle, Mr Rudi Canters [Bli03]. Ce projet belge est soutenu par divers organismes publics dont la Région Wallonne et le Ministère Flamand de l'Egalité des Chances.

- **Statut**

Ce label -et donc les directives qu'il exige pour appliquer le label à un site Internet- est considéré comme la marque de qualité officielle en cours en Belgique, avant l'arrivée d'un label de qualité européen.

- **Nombre de règles**

Les sites web qui, après une évaluation, semblent satisfaire aux éléments obligatoires des **16 directives d'accessibilité** de BlindSurfer peuvent afficher le label sur leur page d'accueil. Les seize directives sont fortement inspirées du WCAG 1.0.

- **Exemples**

NAVIGATION

7. ***Couplage de chaque hyperlien avec un texte significatif***
   Faire en sorte que tout hyperlien qu'il soit graphique ou textuel soit couplé à un texte significatif

- **Avantages/ Inconvénients**

BlindSurfer se base sur les directives du WCAG 1.0. Mais il les interprète à sa façon, avec tout le risque de l'interprétation. L'avantage de BlindSurfer est qu'il met également à disposition des web designer un « Cahier de charges Accessibilité » qui reprend tous les éléments de priorité 1 du WCAG 1.0, indispensable pour l'obtention du label ainsi que d'autres éléments de priorité 2 du WCAG 1.0 pour assurer une accessibilité optimale.

3.2.5 AccessiWeb

De manière similaire, en France, un label de qualité des sites web a été créé.

- **Titre**

C'est l'association BrailleNet qui a créé ce label qui garantit l'accessibilité d'un site. Le titre de ce label est **AccessiWeb** [Acc05].

- **Provenance**

Ce label est créé par l'association BrailleNet [BrN05] fondée en 1997. Cette association a pour objectif de mettre les nouvelles technologies d'information et de communication et l'édition numérique au service de l'intégration sociale et culturelle des aveugles et malvoyants. Internet peut constituer un formidable outil d'intégration et l'association BrailleNet vise à encourager le développement de ce potentiel dans les domaines de l'information, l'éducation et la culture.

- **Statut**

Tout comme BlindSurfer, AccessiWeb est le label de qualité le plus important en France. La citation suivante d'un article de www.zdnet.fr [Per04] illustre l'importance d'AccessiWeb sur le territoire français

« Le gouvernement français travaille actuellement sur un référentiel national, qui serait intégré au cadre commun d'interopérabilité, et aurait ainsi valeur normative pour tout appel d'offre lié à la construction d'un site public. Ces dispositions, comme un certain nombre d'autres, font partie d'un rapport remis récemment à l'administration. De nombreuses adresses sur l'accessibilité des sites […]. On peut citer le site du WAI […], ou AccessiWeb qui est un des sites de l'association BrailleNet, acteur français en la matière. »

- **Nombre de règles**

Ce label existe en 3 versions (Bronze, Argent et Or) et sont attribuées aux sites web qui satisfont à l'évaluation faite par les experts de l'association BrailleNet sur la base d'une grille de **92 critères**. Les critères sont répartis en 3 groupes correspondants aux 3 niveaux de label (55 critères de Bronze, 23 d'Argent et 14 d'Or). Ce regroupement est très semblable à celui des priorités du WCAG 1.0.

- **Exemples**

  4. Multimédia.
  4.1 : Est-il possible de récupérer les informations fournies dans les supports multimédias d'une autre manière ?
  4.2 : Le contenu multimédia est-il synchronisé avec son alternative ?

- **Avantages/ Inconvénients**

L'avantage des 92 critères d'AccessiWeb réside dans leur expression et leur couverture, même si les critères sont largement inspirés de ceux du WCAG 1.0, leur interprétation est pertinente. Cet ensemble de 92 critères est établi à partir des recommandations WCAG 1.0, mais également sur la base de 300 évaluations de site par BrailleNet depuis 1998.

Mais leurs critères sont malheureusement très spécifiques au langage HTML. D'autant plus que les critères sont parfois classés selon les éléments propres au langage HTML.

3.2.6 Usability.gov

Dans notre recherche des règles d'accessibilité, nous en avons trouvé quelques-unes sur le site http://usability.gov géré par l'United States Department of Health & Human Services (HHS) [HHS05]. Ce site fournit une version en ligne et hors ligne (sous format pdf – 128 pages de deux feuilles en vis à vis) d'un ensemble de directives pour la conception de sites web avec un haut degré d'utilisabilité [Koy03].

- **Titre**

Le titre du document est « **Research – Based Web Design & Usability Guidelines** ».

- **Provenance**

L'HHS fournit ce document puisque la communication en ligne claire et efficace est un élément critique dans sa stratégie de communication. D'autant plus que le site a été développé spécifiquement pour aider les gestionnaires, concepteurs et auteurs de site web du HHS à améliorer leurs efforts de conception et de communication.
Les auteurs du document sont Sanjay J. Koyani, Robert W. Balley et Janke R. Nall.
Sanjay J. Koyani est un ingénieur ergonome du département technologie de communications du « National Cancer Institute ». Mr Koyani a dirigé la recherche de l'utilisabilité et développé des outils d'utilisabilité comme Usability.gov pendant les 4 dernières années. Dr. Robert W. Bailey est le président de Computer Psychology, Inc. Il est impliqué dans l'étude de l'ergonomie et de l'utilisabilité depuis plus de 35 ans. Janice R. Nall est la responsable du département technologie de communications du « National Cancer Institute ». Janice a créé le premier groupe d'étude et le premier laboratoire d'utilisabilité du HHS.

- **Statut**

Ce document est soutenu par le « National Cancer Institute », par l'United States Department of Health & Human Services qui sont des organismes importants au Etats-Unis.

- **Nombre de règles**

Le document « Research – Based Web Design & Usability Guidelines » présente ses conseils de conception en 17 différentes catégories, mais seule la catégorie « Accessibility » nous intéresse pour l'instant, les autres catégories nous aiderons pour la collecte des règles d'ergonomie. Sous la catégorie de l'accessibilité, le document reprend **13 directives**.

- **Exemples**

3. Accessibility

3.8 Enable Users to Skip Repetitive Navigation Links

To aid those using assistive technologies, provide a means for users to skip repetitive navigation links.

- **Avantages/ Inconvénients**

Ce document est un travail titanesque qui fait référence à d'innombrables sources, et peut être considéré comme une des plus importantes collations récentes de règles d'ergonomie. L'autre avantage de ce document est l'indication d'un niveau d'importance et d'évidence pour chacune des règles (Nous ne les avons pas reprises dans les annexes, mais le document est disponible en ligne). Par contre au niveau du chapitre 3 concernant l'accessibilité, le document n'apporte pas beaucoup d'interprétation, et est peut-être un peu trop inspiré de la section 508.

3.2.7 Autres sources

Evidement, les sources présentées ne sont pas les seules à parler de règles d'accessibilité. Beaucoup d'organisations nationales, gouvernementales ont défini des règles d'accessibilité pour leur pays, comme l'Irlande [INDA05], l'Angleterre [UKEGU02] ou le Japon [Wata04] Mais presque tous s'alignent sur celles dictées par la WAI.

3.3 Les sources des règles d'ergonomie

Concernant les règles d'ergonomie, nous avons aussi essayé de parcourir les sources le plus largement possible, mais c'est un domaine qui existe depuis l'usage d'interface utilisateur et qui ne se limite pas à l'informatique. Le domaine est nettement plus large que le sous-ensemble des règles d'accessibilité, nous ne pouvons donc pas présenter une liste exhaustive des sources. Les 4 premières sous-sections suivantes constituent nos sources de base. Ce que nous appelons règles d'ergonomie sont souvent davantage des conseils, des heuristiques issus de longues années de pratique, par opposition aux règles d'accessibilité qui sont des recommandations plus strictes de par leurs caractères juridiques dans de plus en plus de pays.

3.3.1 Usability.gov

Le document intitulé « Research – Based Web Design & Usability Guidelines » [Koy03] que nous avons déjà présenté dans la première section, traite également de règles d'ergonomie.

- **Nombre de règles**

Ces règles d'ergonomie qui incluent celle d'accessibilité sont structurées en 17 catégories et sont au nombre de 187. Nous listons les catégories ci-dessous avec, pour chacune d'entre elles, le nombre de recommandations indiquées entre parenthèses.

1. Design Process and Evaluation (16)
2. Optimizing the User Experience (14)
3. Accessibility (13)
4. Hardware and Software (5)
5. The Home Page (9)
6. Page Layout (10
7. Navigation (10)
8. Scrolling and Paging (5)
9. Headings, Titles, and Labels (8)

10. Links	(14)
11. Text Appearance	(7)
12. Lists	(8)
13. Screen-based Controls (Widgets)	(25)
14. Graphics, Images, and Multimedia	(15)
15. Writing Web Content	(11)
16. Content Organization	(9)
17. Search	(8)

- **Exemples**

1. Design Process and Evaluation

1.1 Set and State Goals

Identify and clearly articulate the primary goals of the website before beginning the design process.

6. Page Layout

6.4 Place Important Items at Top Center

Put the most important items at the top Relative Importance: center of the Web page to facilitate users' finding the information.

- **Avantages/ Inconvénients**

Les catégories couvrent une grande partie du processus de conception, elles incluent par exemple des recommandations sur la conception des idées, la communication entre les développeurs et non uniquement sur le code, ou le rendu final du site web ou interface. De même ces règles sont parfois très abstraites dans le sens où elles sont parfois plus générales et moins liées à une technologie comme nous l'observons pour les règles d'accessibilité soumises par la WAI.

3.3.2 The Essential Guide to User Interface Design.

Une autre des sources très intéressantes que nous avons parcourues, est le livre de Wilbert O. Galitz.

- **Titre**

Le titre du livre est « The Essential Guide to User Interface Design: An introduction to GUI design principles and techniques »[Gal96]. Ce livre fut publié une première fois en 1996 et en une seconde édition en 2002. Il s'adresse au concepteur et programmeur de systèmes qui ont besoin de créer des interfaces graphiques ergonomiques « User-Friendly ».

- **Provenance**

Son auteur est Wilbert O. Galitz. Il est de renommée internationale en tant que consultant, auteur et instructeur dans les systèmes informatiques. Il est également le président de Galitz, Inc., une société qui se concentre sur les facteurs humains et l'ergonomie des systèmes.

- **Statut**

Ce livre donne une très bonne introduction complète à la conception des interfaces graphiques et il est très souvent cité dans d'autres livres comme une référence concernant la conception des interfaces graphiques.

- **Nombre de règles**

Ce livre contient *des centaines de principes de bonne conception d'interfaces graphiques.* Les centaines de règles non numérotées sont triées en 12 étapes définissant le processus de conception des interfaces graphiques. Avec un langage concis et des illustrations détaillées, le livre guide le lecteur, d'étapes en étapes, à travers le processus de conception, en montrant ce qu'il faut et ce qu'il ne faut pas faire avec des exemples de bonnes et mauvaises conceptions d'interface..

- **Exemples**

Simplicity (Complexity)

- *Optimize the number of elements on a screen, within limits of clarity*
- *Minimize the alignement points, especially horizontal or columnar*
  - *Provide standard grids of horizontal and vertical lines to position elements.*

- **Avantages/ Inconvénients**

On peut cependant déjà souligner quelques références dépassées que possède le livre dans sa première version. Certains principes d'ergonomie contenus dans le livre considèrent une interface à « caractères », avec des recommandations du type : *« separate the symbol from the heading by one space and from the captions by a minimum of three spaces. »*. En effet, sur ce type d'interface, les espaces sont des caractères blancs. Actuellement, on parlera plus en pixel ou en centimètres, ou plus du tout, lorsque l'on fait usage de toolkits graphiques qui incluent eux-mêmes certaines contraintes de présentations.

3.3.3 De l'ergonomie du logiciel au design des sites web.

Le second livre que nous avons consulté lors de la recherche des règles d'ergonomie est celui de Jean-François Nogier.

- **Titre**

Le titre du livre est « De l'ergonomie du logiciel au design des sites web » [Nog02]. Il est paru en 2002. Ce livre tente de faire la synthèse des règles issues des différentes études et expérimentations menées au cours des dernières années dans le domaine de l'ergonomie du logiciel. C'est-à-dire qu'il présente des méthodes et donne des conseils pratiques afin de concevoir des logiciels qui seront utilisés efficacement et agréablement. Tout comme le livre de Galitz, il s'adresse à toutes personnes impliquées dans la conception et le développement logiciel, plus particulièrement des interfaces homme-machine, et donc inévitablement aux développements de sites Internet, puisque ces derniers sont principalement des interfaces homme-machine.

- **Provenance**

Son auteur, Jean-François Nogier est docteur ingénieur en informatique. Sa carrière s'est essentiellement consacrée au domaine des interfaces homme-machine. En 2001, il fonde Usabilis un cabinet de conseil spécialisé en ergonomie informatique. Il enseigne également l'utilisabilité du logiciel à l'université de Paris Dauphine, à l'Institut National des Télécommunications (INT) et à l'Institut Supérieur d'Électronique de Paris (ISEP)

- **Statut**

Ce livre est particulièrement reconnu comme référence par les auteurs francophones du domaine. Comme en témoigne les critiques et points de vue suivants:

- « *L'ouvrage de Jean-François Nogier est de ceux qui expliquent simplement des idées complexes. De ceux à ne pas rater.* » Amélie Boucher (27/05/04)
- « *Cet ouvrage de référence est devenu un incontournable !* » Michelle Sincennes, Biosphère d'Environnement Canada (07/02/04)
- « *Un livre qui devrait figurer sur toutes les tables de chevet des spécialistes en utilisabilité* » Jérôme Ernu (22/10/03)
- « *A recommander à tous les graphistes* » Association d'Ergonomie d'Orsay (15/01/03)

- **Nombre de règles**

Il est impossible de donner une indication sur le nombre de règles, tant les recommandations sont dispersées dans le livre et le fait qu'elles sont plus des conseils donnés au cours de la lecture du livre par un développeur qu'un recueil de règles comme l'est le document « Research – Based Web Design & Usability Guidelines » précité.

- **Exemples**

Ne pas changer la couleur et le comportement par défaut des liens.
Les informations affichées doivent être non ambiguës et faciles à lire

Produire des messages brefs, concis et pertinents.

- **Avantages/ Inconvénients**

Ce livre présente les principes d'ergonomie ainsi que de très bons exemples d'applications de ces principes pour une technologie en particulier, celle des sites web. Cela permet d'interpréter de manière plus précise certains principes d'ergonomie relativement abstraits. L'inconvénient est que les règles ne sont pas suffisamment structurées, numérotées.

3.3.4 Autres sources

Nous n'avons évidement pas pu compulser tous les livres parlant d'ergonomie afin de collecter les différentes recommandations. Mais nous pouvons, par contre, signaler la présence d'un nombre incroyable d'ensembles de règles proposées presque chaque année par un nouvel auteur.

En 1986, Sidney L. Smith et Jane N. Mosier publient un livre de 500 pages intitulé « Guidelines for designing user interface software » contenant 944 recommandations [Smi86]c'est à notre connaissance le livre contenant le plus de recommandations. Cet

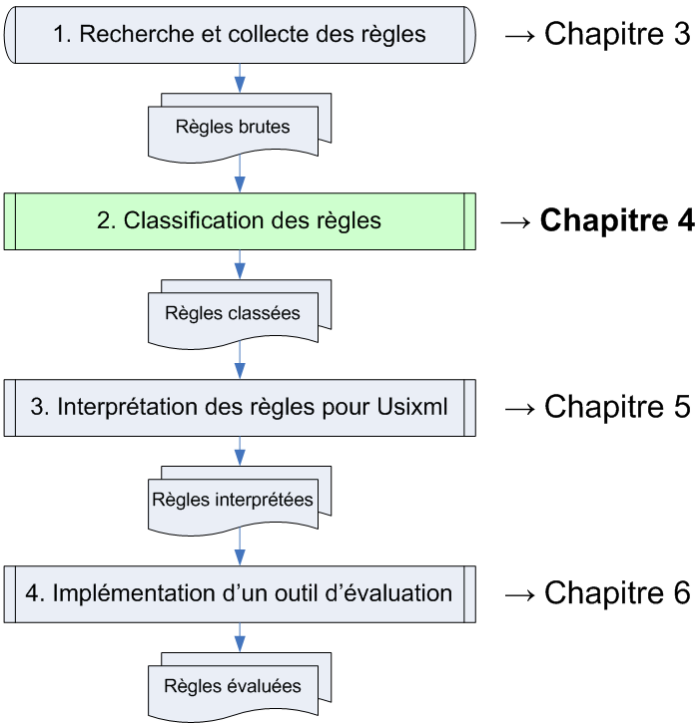
ouvrage de la MITRE Corporation, Bedford, Massachusetts est rendu public et est actuellement disponible en ligne à l'adresse suivante : http://www.hcibib.org/sam/ .
Un an après la sortie du livre de Smith et Mosier, Marshall participe à la rédaction d'un chapitre intitulé « Design guidelines » pour le livre « Applying cognitive psychology to user-interface design » de Margaret M. Gardiner et Bruce Christie. Dans ce chapitre seront réunies 162 recommandations [Mar87]. Un an après encore, C. Marlin Brown sort un livre de 302 recommandations : « Human-Computer Interface Design Guidelines » [Bro88]. Un autre ouvrage, publié en 1992 et reprenant 288 recommandations est celui de Deborah J. Mayhew « Principles and guidelines in software user interface design »[May92].

Le nombre de recommandations varie d'un ouvrage à l'autre dans de grandes proportions. Il n'est pas étonnant que ces directives aient été critiquées pour être à la fois trop générales et trop spécifiques [Con00]. Elles sont trop générales parce qu'il est difficile de traduire leur contenu en conseil de conception pour un système spécifique et de donner la priorité entre les directives. Elles sont trop spécifiques quand elles se concentrent sur la technologie courante aux dépens de conseils plus généraux. De plus, la taille de certaines collections peut également rendre difficile la localisation de recommandations spécifiques [Smith and Mosier].

Chapitre 4 Classification des règles

Dans le chapitre précédent, nous avons présenté différentes sources, soit différents ensembles de règles. Chacune des sources d'accessibilité est d'ailleurs reprise en annexe. Nous sommes à la seconde étape de notre méthodologie. Les règles sont dites « brutes ». Nous allons tenter de les organiser, de les classifier pour produire un ensemble de règles classifiées.

Dans ce chapitre, nous présenterons la classification des règles d'accessibilité dans la section 0, pour ensuite faire de même pour les règles d'ergonomie dans la section 4.2. Lorsque les règles seront classifiées, organisées, il nous sera dès lors possible de les interpréter pour un langage particulier. Il en sera question dans le chapitre suivant.



Notre méthode de classification est constituée de deux processus. Le premier consiste à trouver une règle générale ou largement reprise par l'ensemble des sources et à présenter l'idée générale. Le deuxième processus consiste à regrouper les règles parlant du même sujet, de la même idée -ou a peu près- en un ensemble représenté dans des tableaux.

4.1 Classification des règles d'accessibilité

Pour les règles d'accessibilité, afin de ne pas trop dépendre du choix technologique des différentes sources, nous avons modifié légèrement la classification en 4 principes de base du WCAG 2.0 qui est toujours en développement. Au lieu des 4 critères du WCAG 2.0, nous n'en gardons que 3, le quatrième principe : « Robuste » concerne l'adaptation aux technologies courantes et futures. Mais puisque nous nous ne voulons pas nous baser de trop sur la technologie, toutes les règles qui devraient se retrouver sous ce critère ont été supprimées ou reclassées selon leur pertinence. Notre classification est très simple et correspond à deux principes de base, le contenu et la navigation.

- **Le contenu** : c'est-à-dire que le contenu doit être accessible à tous. Ce premier critère à été séparé en deux pour attirer l'attention sur deux idées importantes reprises par le WCAG également, la perception et la compréhension.
    - **La perception** : c'est-à-dire la capacité qu'a l'utilisateur à percevoir l'information.
    - **La compréhension** : c'est-à-dire la capacité qu'a l'utilisateur de comprendre l'information qu'il perçoit.
- **L'utilisation ou la navigation** : c'est-à-dire que le contrôle de l'interface doit être accessible.

Pour chaque critère, nous donnerons une courte explication, suivie de quelques recommandations issues des différentes sources que nous avons regroupées. Les recommandations sont diverses et tendent à couvrir des aspects parfois différents tout en étant spécifiques.

4.1.1 Contenu – Perception

Ce critère signifie que le contenu doit être accessible à toute personne et doit être perçu par tous. Cela inclut toutes les règles relatives aux versions alternatives des objets qui ne peuvent être perçus directement. Cela concerne aussi la mise en page et la présentation du contenu, par exemple, les couleurs.
Afin d'illustrer ce critère, nous allons prendre quelques exemples de règles qui se retrouvent dans les différentes directives.

4.1.1.1 Premier regroupement

Prenons la toute première directive du WCAG, qui est aussi sans doute la plus connue.

> *Guideline 1 : « Provide equivalent alternatives to auditory and visual content ».*
> *1.1 Provide a text equivalent for every non-text element.*
> *1.2 Provide redundant text links for each active region of a server-side image map.*

> *1.3 Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation.*
> *1.4 For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation.*
> *1.5 Until user agents render text equivalents for client-side image map links, provide redundant text links for each active region of a client-side image map.*

Cette recommandation, qui se retrouve un peu partout dans les autres sources (voir tableau de correspondance.) correspond clairement au critère de perception du contenu. L'intérêt de cette règle vient de l'usage d'aides techniques tel que les screen readers vocaux (IBM Home Page Reader [Ibm05c]) ou les tablettes Braille qui ne transmettent que les textes. En effet, ces aides techniques utilisent le même principe que les navigateurs textuels tel que Lynx [Dic04] ou BrailleSurf [Sch01].

Cette directive du WAI, n'est pas typique à la conception des sites web. Les directives d'IBM pour l'accessibilité des logiciels traduisent cette idée en deux recommandations, l'une pour l'information des objets en général et l'autre pour le contenu auditif et audio-visuel.

> *Provide semantic information about user interface objects. When an image represents a program element, the information conveyed by the image must also be available in text.*

De manière identique, la section 508 fait de même avec la recommandation suivante.

> *Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.*

Sources	**Directives**
WCAG 2.0	1.1 [L1, L3], 1.2 [L1, L2, L3], 4.2[L1, L2, L3]
WCAG 1.0	1.1, 1.2, 1.3, 1.4, 1.5, 3.1, 6.2, 6.3, 6.5, 8.1, 9.1, 11.4
AccessiWeb	1.1, 1.2, 1.3, 1.4, 1.5, 1.8, 1.9, 1.10, 1.11, 4.1, 4.2, 7.1, 11.6
BlindSurfer	1, 3, 5, 10, 11, 16
IBM – Web	1, 2, 3, 4, 5, 6, 15
IBM – Software	2.2, 3.1, 3.2
IBM – Java	2.1, 3.1, 3.2
508 – Web	a, b, e, f, k, l
508 – Software	d, e, f
Usability.gov	3.3, 3.5, 3.6, 3.7, 3.10, 3.11

4.1.1.2 Second regroupement

Une autre idée générale reprise par les différentes directives est celle définie dans WCAG 2.0 sous le principe 1.3.

> *Guideline 1.3 Ensure that information, functionality, and structure can be separated from presentation.*
> *Level 1 Success Criteria for Guideline 1.3*
> - *Structures within the content can be programmatically determined.*
> - *When information is conveyed by color, the color can be programmatically determined or the information is also conveyed through another means that does not depend on the user's ability to differentiate colors.*
>
> *Level 2 Success Criteria for Guideline 1.3*
> - *Information that is conveyed by variations in presentation of text is also conveyed in text or the variations in presentation of text can be programmatically determined.*
> - *Any information that is conveyed by color is visually evident when color is not available.*
>
> *Level 3 Success Criteria for Guideline 1.3*
> - *When content is arranged in a sequence that affects its meaning, that sequence can be determined programmatically.*

Cette directive, très large, est reprise dans les autres sources d'une manière souvent plus spécifique, particulièrement pour les sites web. L'idée générale est qu'il faut séparer la structure, les fonctionnalités et l'information de la façon dont ce sera présenté. En effet, on ne sait pas quel sera la plate-forme sur lequel l'interface tournera. Un grand ou petit écran, un écran couleur ou monochrome. Cette directive prévient la situation où l'information qui serait incluse dans la présentation ne puisse être rendue accessible à l'utilisateur si ce dernier utilise une plate-forme qui modifie la présentation ou restreint les capacités de présentation. [Bra05a][Bra05b][Her98]

Quelques exemples de spécification de cette directive générale pour le web nous aident à mieux comprendre sa signification.
Le WCAG 1.0 spécifie cette directive pour les tables en plusieurs directives :

> *5.1 For data tables, identify row and column headers.*
> *5.2 For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells.*
> *5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting.*
> *5.5 Provide summaries for tables.*

Nous ne retrouvons pas, telle quelle, cette directive générale autre part que pour le web, excepté concernant la couleur qui ne doit pas fournir d'information à elle seule. Et cela semble assez logique dans le sens où les sites web ont la particularité d'être des interfaces dont la présentation est interprétée par les navigateurs, par opposition aux logiciels compilés dont la présentation est figée dans une certaine mesure. Cela dit, parmi les

recommandations spécifiques pour Java et les logiciels qu'IBM nous proposent, les deux suivantes sont assez proches du but recherché.

> *Implement the Java Accessibility API by:*
> *- using the Java Foundation Classes (JFC) / Swing components and/or*
> *- following the guidelines for "Building Custom Components" when extending the Java Foundation Classes and when implementing the Java Accessibility API on custom components.*
>
> *Inherit system settings for font, size, and color for all user interface controls.*

Sources	**Directives**
WCAG 2.0	1.3 [L1, L2, L3]
WCAG 1.0	2.1, 3.3, 3.4, 3.5, 3.6, 3.7, 5.1, 5.2, 5.3, 5.4, 5.5, 6.1
AccessiWeb	3.1, 5.1, 5.2, 5.3, 5.4, 10.1, 10.2, 10.3, 10.4, 10.5, 10.6, 13.8,
BlindSurfer	2, 4, 13
IBM – Web	10, 11, 12
IBM – Software	4.1, 4.2, 4.5
IBM – Java	2.1, 4.1, 4.4
508 – Web	c, d, g, h
508 – Software	f, i
Usability.gov	3.4, 3.12

4.1.1.3 Troisième regroupement

On ne peut évidement pas aborder le sujet de la perception d'une interface sans parler du problème occasionné par les couleurs. [New00]. En effet, près de 8% des hommes et 0.4% des femmes sont atteints d'une déficience de vision des couleurs. C'est une proportion de la population non négligeable. Pour ces personnes, les interfaces doivent présenter un contraste entre les couleurs de l'interface, particulièrement entre les informations d'avant plan et d'arrière plan. La plupart des sources sont d'accord sur le problème du contraste des couleurs. Ce souci peut être repris sous une directive plus générale telle que le propose le WCAG 2.0 :

> *Make it easy to distinguish foreground information from background images or sounds.*

Ou, d'une manière plus spécifique pour le web, comme le propose AccessiWeb :

> *Les différences de contrastes entre les couleurs sont-elles suffisamment élevées ?*

Notons qu'il existe sur le net, un nombre important d'outils permettant de simuler les déficiences visuelles et de choisir correctement les couleurs pour valider ces recommandations [Sno05][CSG05]

Sources	Directives
WCAG 2.0	1.4 [L1, L2, L3]
WCAG 1.0	2.2
AccessiWeb	3.2
BlindSurfer	4, 12
IBM – Software	4.3, 4.4
IBM – Java	4.2, 4.3
508 – Software	g , j

4.1.2 Contenu – Compréhension

Ce critère signifie que le contenu doit être compréhensible, lisible par tous. Ce critère est différent du premier dans le sens où la perception ne permet pas nécessairement la compréhension. Les recommandations qui sont classées sous ce critère concernent principalement le langage, comme les abréviations par exemple, la facilité d'apprentissage, comme la consistance dans la structure de l'interface. La facilité de compréhension et la consistance sont des critères ergonomiques. On se rend bien compte que les règles d'accessibilité et d'ergonomie sont étroitement liées. Comme nous l'avons déjà dit, les règles d'ergonomie sont utilisées pour optimiser l'usage de l'interface par l'utilisateur. Optimiser l'usage en la rendant plus facile d'emploi à un utilisateur moyen, c'est aussi aider un utilisateur victime d'un handicap.

4.1.2.1 Premier regroupement

Une des recommandations que l'on classe sous cette catégorie, est celle de la WAI concernant l'indication de la langue dans les documents. L'indication de la langue permet aux aides techniques vocales de prononcer correctement. Il en est de même avec les abréviations, car sinon les « screen reader » ne sauront pas les prononcer correctement. De plus, indiquer l'extension de chaque abréviation permet à l'utilisateur de savoir de quoi on parle plus facilement que par le contexte. En effet, UCL signifie « Université Catholique de Louvain », et signifie aussi « University College London ».

> *Guideline 4: « Clarify natural language usage »*
> *Use markup that facilitates pronunciation or interpretation of abbreviated or foreign text.*
> *4.1 Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions).*
> *4.2 Specify the expansion of each abbreviation or acronym in a document where it first occurs.*
> *4.3 Identify the primary natural language of a document.*

Ce genre de recommandations ne sont reprises ni par la section 508, ni par IBM, ni par Usability.gov

Sources	Directives
WCAG 2.0	3.1 [L1, L2]
WCAG 1.0	5.6, 4.1, 4.2, 4.3
AccessiWeb	5.5, 8.2, 8.7, 13.9, 13.10, 13.11, 13.12
BlindSurfer	6

4.1.2.2 Second regroupement

Comme nous le voyons dans le critère de succès de niveau 3 de la recommandation 3.1 du WCAG 2.0, il est également question de fournir des aides supplémentaires pour aider à comprendre le contenu. Toutes les personnes sont aidées par ce type de recommandations, mais spécialement les personnes dyslexiques, jeunes ou ayant de la difficulté à réfléchir. Le but de la recommandation suivante est clairement d'aider à la compréhension du contenu du site.

> *Guideline 3.1 Make text content readable and understandable.*
> *Level 3 Success Criteria for Guideline 3.1*
> - *A mechanism is available for finding definitions for all words in text content.*
> - *A mechanism is available for identifying specific definitions of words used in an unusual or restricted way, including idioms and jargon.*
> - *A mechanism for finding the expanded form of acronyms and abbreviations is available.*
> - *Section titles are descriptive.*
> - *When text requires reading ability at or above the upper secondary education level, one or more of the following supplements is available:*
>   - *A text summary that requires reading ability no higher than primary education level.*
>   - *Graphical illustrations of concepts or processes that must be understood in order to use the content.*
>   - *A spoken version of the text content.*

Encore une fois, cette règle n'est pas fort soutenue par les autres sources que nous avons identifiées. Cela peut se comprendre dans le sens où ce type de recommandation est particulier et semble sortir de l'ensemble des règles d'ergonomie et d'accessibilité, parce qu'elle suggère d'ajouter de l'information, alors que généralement ce sont des règles destinées à rendre accessible l'information déjà présente.

Sources	Directives
WCAG 2.0	3.1 [L3]
WCAG 1.0	14.1, 14.2

4.1.2.3 Troisième regroupement

Le troisième regroupement que nous réalisons concerne les règles qui font référence à la consistance de l'interface, afin de la rendre prédictible et ainsi plus compréhensible, plus facile à employer. IBM exprime cette idée dans une recommandation pour les logiciels:

> *Associate labels with controls, objects, icons and images. If an image is used to identify programmatic elements, the meaning of the image must be consistent throughout the application.*

La consistance des liens et de leur description comme le demande BlindSurfer, un style de présentation constant à travers le site web, à travers l'interface utilisateur, contribuent à rendre l'interface plus compréhensible, plus simple aussi.

Sources	**Directives**
WCAG 2.0	3.2 [L1, L2, L3]
WCAG 1.0	10.1, 13.1, 13.4, 14.3
AccessiWeb	1.6, 1.12, 1.13, 5.6, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 9.2, 9.5, 9.7, 11.8, 12.1, 12.2, 12.6, 13.3, 13.4, 13.13
BlindSurfer	7, 14
IBM – Software	2.3
IBM – Java	2.3

4.1.3 Navigation – Opérabilité des objets, de l'interface

Au niveau de la navigation, le souci se situe sur l'opérabilité des objets de l'interface. L'interface doit être utilisable par toute personne. C'est-à-dire que les objets de contrôle doivent pouvoir être accessibles, compréhensibles, adaptables, etc.

4.1.3.1 Premier regroupement

Prenons par exemple les formulaires. C'est l'un des points les plus intéressants d'Internet, puisque c'est le style d'interaction préféré pour toutes les communications avec les magasins, institutions publiques, etc.… Les formulaires doivent pouvoir être utilisables pour des personnes victimes d'un handicap moteur par exemple, donc utilisables à partir du clavier uniquement. De manière générale, toutes les interfaces doivent être accessibles par n'importe quel type d'outils de commande, souris, clavier, voix, mouvement de la tête ou des mains, ou autres encore. C'est dans cette idée générale que le WCAG 1.0 exprime cette idée.

> *Guideline 9: « Design for device-independence »*
> *Use features that enable activation of page elements via a variety of input devices.*
> *9.1 Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape.*

> *9.2 Ensure that any element that has its own interface can be operated in a device-independent manner.*
> *9.3 For scripts, specify logical event handlers rather than device-dependent event handlers.*
> *9.4 Create a logical tab order through links, form controls, and objects.*
> *9.5 Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls.*

Généralement, les pages qui autorisent l'interaction via le clavier sont aussi accessibles à travers la voix et les commandes en ligne grâce aux aides techniques.
L'utilisation de raccourci clavier, d'accélérateurs, de tabulations fait partie également des règles d'ergonomie qui soulignent l'importance de donner aux utilisateurs expérimentés un moyen d'interaction plus rapide que celui de la manipulation directe (via la souris par exemple).

Sources	**Directives**
WCAG 2.0	2.1 [L1, L3]
WCAG 1.0	6.4, 9.2, 9.3, 9.4, 9.5
AccessiWeb	7.2, 11.7, 12.5
BlindSurfer	8
IBM – Web	5
IBM – Software	1.1, 1.2
IBM – Java	1.1, 1.2
508 – Software	a, b

4.1.3.2 Second regroupement

Rendre la navigation opérable c'est aussi donner l'impression à l'utilisateur qu'il contrôle l'interface et non le contraire. En effet, beaucoup de personnes victimes d'un handicap sont plus lentes à la réaction, à la compréhension, à la perception. Des taux de rafraîchissements trop élevés, des textes déroulants trop rapides, des temps de réponse attendus trop court, rendent l'utilisation de l'interface impossible pour ces groupes de personnes. La section 508 l'a bien compris et l'exprime dans ses recommandations pour les sites web.

> *p. When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.*

Tout comme IBM le recommande aussi pour les logiciels.

> *5.1 Provide an option to adjust the response times on timed instructions or allow the instructions to persist.*

Sources	**Directives**
WCAG 2.0	2.2 [L1, L2, L3]
WCAG 1.0	7.4, 7.5
AccessiWeb	13.1, 13.2

IBM – Web	14
IBM – Software	5.1
IBM – Java	5.1
508 – Web	p

4.1.3.3 Troisième regroupement

De manière très similaire au regroupement précédent, beaucoup de recommandations font référence au danger du clignotement de l'écran, des animations intempestives qui peuvent provoquer soit des crises d'épilepsie chez certains groupes de personnes, soit un énervement compréhensible chez d'autres, soit encore une distraction pour les personnes ayant de la difficulté à se concentrer.

Sources	**Directives**
WCAG 2.0	2.2 [L3], 2.3 [L1, L2, L3]
WCAG 1.0	7.1, 7.2, 7.3
AccessiWeb	13.7
IBM – Web	13
IBM – Software	5.2, 4.6, 3.3
IBM – Java	5.2, 4.5, 3.3
508 – Web	j
508 – Software	h, k
Usability.gov	3.13

4.1.3.4 Quatrième regroupement

Evidement, la navigation ne peut exister que si l'interface donne la possibilité de naviguer, c'est-à-dire de donner des mécanismes pour trouver l'information, s'orienter et naviguer à travers l'interface.

On recommande par exemple de donner un nom à tous les objets de l'interface, aux fenêtres etc., de fournir des moyens de navigation tel que des menus ou des barres de navigation.

De même, il est nécessaire d'indiquer le focus courant, ainsi les personnes aveugles peuvent écouter quel est le focus courant grâce aux aides techniques et ainsi reprendre le travail après une brève interruption.

Sources	**Directives**
WCAG 2.0	2.4 [L1, L2, L3] 2.5 [L1, L2, L3]
WCAG 1.0	10.2, 10.3, 10.4, 10.5, 12.1, 12.2, 12.4, 12.3, 13.2, 13.3, 13.5, 13.6, 13.7, 13.8, 13.9, 13.10
AccessiWeb	1.7, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 2.10, 8.3, 8.4, 8.5, 8.6, 9.3, 9.4, 9.6,

	9.8, 11.1, 11.2, 11.3, 11.4, 11.5, 12.3, 12.4, 12.7, 12.8
BlindSurfer	9, 15
IBM – Web	6, 7, 8, 9
IBM – Software	2.1, 2.2, 2.3, 2.4
IBM – Java	2.2, 2.3, 2.4
508 – Web	i, m, n, o
508 – Software	c, d, l
Usability.gov	3.2, 3.8, 3.9

4.1.4 Autres règles

Pour compléter notre classification, nous présentons ici les quelques règles qui nous ont semblé soit trop spécifiques ou non pertinentes.

Prenons par exemple la première recommandation concernant l'accessibilité que nous propose le site Usability.gov :

> *3.1 « Comply with Section 508 » : If a website is being designed for the United States government, ensure that it meets the requirements of Section 508 of the Rehabilitation Act. Ideally, all websites should strive to be accessible and compliant with Section 508.*

Cette recommandation, similaire à la onzième directive du WAI, ne fait que recommander de suivre les recommandations.

Sources	**Directives**
WCAG 2.0	4.1
WCAG 1.0	3.2, 11.1, 11.2, 11.3
AccessiWeb	8.1, 13.5, 13.6
IBM – Web	16
IBM – Software	6.1
IBM – Java	6.1, 6.2, 7.1
Usability.gov	3.1

4.2 Classification des règles d'ergonomie

Pour classer les règles d'ergonomie, nous avons choisi une classification sur base des critères développés à l'INRIA (1993) par J.C. Bastien et Scapin [Bas93]. Ces critères sont largement acceptés et sont d'ailleurs repris dans la norme AFNOR Z67-133-1.
On retrouve à peu près ces mêmes critères dans toutes les études du domaine, que ce soit dans les règles d'évaluation heuristique de Jakob Nielsen [Nie93] ou dans les cours d'Interface Homme-machine (IHM) donnés à l'Université Catholique de Louvain [Vand05].

Les critères ergonomiques représentent un moyen de classification des recommandations, mais ils représentent surtout les dimensions ergonomiques majeures selon lesquelles une interface utilisateur peut être évaluée. Les critères ergonomiques se subdivisent en sous critères comme illustré ci-dessous.

1. Guidage
1.1 Incitation
1.2 Groupement/Distinction entre items
1.2.1 Groupement/Distinction par la localisation
1.2.2 Groupement/Distinction par le format
1.3 Feed-back immédiat
1.4 Lisibilité

2. Charge de travail
2.1 Brièveté
2.1.1 Concision
2.1.2 Actions minimales
2.2 Densité informationnelle

3. Contrôle explicite
3.1 Actions explicites
3.2 Contrôle utilisateur

4. Adaptabilité
4.1 Flexibilité
4.2 Prise en compte de l'expérience de l'utilisateur

5. Gestion des erreurs
5.1 Protection contre les erreurs
5.2 Qualité des messages d'erreur
5.3 Correction des erreurs

6. Homogénéité/Cohérence

7. Signifiance des codes et dénominations

8. Compatibilité

Il n'est pas possible de classifier de manière exhaustive toutes les recommandations issues des sources présentées au chapitre 3. C'est pourquoi nous présenterons les différents critères d'ergonomie, ainsi qu'une explication sur leur correspondance vis-à-vis des 5 propriétés (*apprentissage, efficacité, mémorisation, traitement de l'erreur et satisfaction*) de l'interface qu'englobe le concept d'utilisabilité [Nie93] que les Anglo-Saxons utilisent plus volontiers, suivie de quelques recommandations d'ergonomie illustrant l'idée.

4.2.1 Guidage

Le guidage est l'ensemble des moyens mis en œuvre pour conseiller, orienter, informer et conduire l'utilisateur lors de ses interactions avec l'ordinateur (messages, alarmes, labels, etc.), y compris dans ses aspects lexicaux [Bas98] , c'est-à-dire lui faire connaître l'état du système et lui permettre d'établir des liens de causalité entre ses actions et l'état du système.

Ce critère, dont l'objectif est de faciliter l'utilisation de l'interface et son apprentissage, augment la capacité d' « apprentissage » puisqu'il informe l'utilisateur à chacune des étapes ou actions de ce qu'il faut faire, augmente le « traitement d'erreur » puisque l'utilisateur est un peu forcé à suivre une séquence de tâches, ainsi l'utilisateur ne risque pas de manquer des choses importantes, mais par contre cela diminue l' « efficacité » dans le sens où les utilisateurs sont forcés de suivre le guide.

Le premier sous critère est **l'incitation**, qui réunit les moyens visant à conduire l'utilisateur à effectuer des actions spécifiques. C'est sans doute le plus important pour la conception des sites web.

> *Voir les règles **5.2**, **5.7** et **6.4** de «Research – Based Web Design & Usability Guidelines » dans les annexes.*
>
> *L'étape 10 du livre Essential guide to user interface design [Gal96] (page540)*

Le second sous critère est le **groupement/distinction entre items**. L'utilisateur qui utilise un outil pour la première fois, applique la règle de la similarité. C'est-à-dire qu'il prend en compte la topologie (localisation) et certains critères graphiques (format) afin de comprendre les relations entre les informations affichées. Il supposera que les objets similaires fonctionneront de manière similaire.

Règle générale : Distinguer par une présentation (format, position) respectivement différente (identique) des informations distinctes (similaires).

> *Regrouper les informations en relation entre elles.*
> *Utiliser des couleurs très contrastées pour exprimer une différence.*
> *Choisir des couleurs peu contrastées pour exprimer une similarité. [Nog02]*

> *Voir les règles **6.6** et **7.4** de «Research – Based Web Design & Usability Guidelines » dans les annexes.*

Le troisième sous critère est le **retour utilisateur** (feedback). Ce sous critère réunit les différents comportements de l'interface indiquant le fonctionnement du système. Cela contribue à accroître la « confiance » de l'utilisateur dans le système.

Règle générale : l'interface doit répondre à toute action de l'utilisateur par un changement de sa présentation. Toujours faire apparaître les saisies de l'utilisateur.

> *Voir les règles **2.10, 2.11** et **7.1** de «Research – Based Web Design & Usability Guidelines » dans les annexes. [Nog02]*

Le dernier sous critère est la **lisibilité**. Ce critère tient compte des caractéristiques matérielles de présentation des informations qui doivent en faciliter la lecture. C'est-à-dire la typographie, l'espacement, etc. Ce critère facilite la perception des informations, donc aussi au guidage. Notons que la perception des informations est un de nos critères d'accessibilité.

> *Les textes doivent être affichés en minuscules, la première lettre étant en majuscule. [Nog02]*

> *Voir la règle **5.8** de «Research – Based Web Design & Usability Guidelines » dans les annexes.*

4.2.2 Charge de travail

La charge de travail concerne l'ensemble des éléments de l'interface jouant un rôle dans la réduction de la charge perceptive et mnésique des utilisateurs. Minimiser la charge cognitive permet d'améliorer la capacité de « traitement d'erreur » et l' « efficacité » puisque l'utilisateur est moins distrait par des objets ou fonctions qui risquent de lui faire commettre des erreurs ou qui ne l'intéressent pas.
Il y a deux sous critère, celui de la **brièveté : Concision / Actions minimales** et celui de **densité informationnelle**.

Le sous critère de **brièveté** s'intéresse à n'afficher que les informations pertinentes et à réduire le nombre d'actions élémentaires pour atteindre un objectif donné.
Voici quelques exemples de recommandations concernant la brièveté.

> *Liste d'items courts, libellés concis, clic distance, minimiser les contraintes de défilement des pages. [Nog02]*

Le second sous critère, **densité informationnelle**, concerne principalement la charge de travail du point de vue perceptif et mnésique pour des ensembles d'éléments. Et non pour des items comme le prend en charge le premier sous critère.

Nombre de liens excessifs, pas plus de 8 liens par exemple. [Nog02]

4.2.3 Contrôle explicite

Ce critère concerne à la fois la prise en compte par le système des actions explicites des utilisateurs et le contrôle qu'ont les utilisateurs sur le traitement de leurs actions. Ce critère indique que l'utilisateur devrait avoir l'impression d'avoir le contrôle de l'application. Cela renforce la « satisfaction » que l'utilisateur a de l'interface, et cela diminue aussi le risque d'erreur puisque l'utilisateur contrôle mieux ce qu'il veut faire communiquer au système. Le premier sous critère est celui de l'**action explicite**, c'est-à-dire que le système doit exécuter uniquement les opérations demandées par l'utilisateur et pas d'autres opérations.

Le second sous critère est le **contrôle utilisateur**, c'est-à-dire que l'utilisateur doit toujours avoir la main, pouvoir contrôler le déroulement (interrompre, reprendre) des traitements informatiques. Cela inclut d'avoir la possibilité d'interrompre une animation, de revenir en arrière, d'annuler ou de refaire sa dernière opération.

4.2.4 Adaptabilité

L'adaptabilité d'un système concerne sa capacité à réagir selon le contexte et selon les préférences des utilisateurs. En effet, l'interface est destinée à être utilisée par différentes personnes susceptibles de travailler différemment, selon leur préférence.

L'adaptabilité se divise en deux sous critères également, celui de la **flexibilité** et celui de la **prise en compte de l'expérience de l'utilisateur**.

Le premier sous critère, la **flexibilité**, se définit comme les moyens mis à la disposition des utilisateurs pour personnaliser l'interface afin de rendre compte de leurs stratégies ou habitudes de travail et des exigences de la tâche [Bas98]. Cela concerne évidement les transformations possibles pour les personnes handicapées comme les règles d'accessibilité que nous avons classifiées, mais également les habitudes des autres utilisateurs comme par exemple la possibilité du logiciel Winamp à utiliser différents skins de présentation [Nul05].

Permettre d'activer les commandes à la fois au clavier et à la souris.
Permettre à l'utilisateur de paramétrer le logiciel selon ses préférences. [Nog02]

Le second sous critère, **prise en compte de l'expérience de l'utilisateur**, correspond plus à la capacité des applications Office à cacher les éléments de menu les moins utilisés, rendant ainsi la charge de travail plus faible également, par exemple. Il est préférable par exemple de guider les novices pas à pas, mais de donner aussi la possibilité aux experts d'aller plus vite.

Fournir un moyen rapide d'accéder aux commandes des menus. [Nog02]

4.2.5 Gestion des erreurs

La gestion des erreurs concerne tous les moyens visant à éviter et réduire les erreurs utilisateurs, ainsi que les corriger dès qu'elles surviennent, afin de conserver l'intégrité de l'application. Les erreurs sont des saisies de données incorrectes, des saisies dans des formats inadéquats, des saisies de commandes avec une syntaxe incorrecte, etc.…
Cela augmente la confiance de l'utilisateur dans l'interface puisqu'il fait moins de fautes, c'est-à-dire augmente la capacité de « traitement d'erreur » tel que le définit J. Nielsen. Cela améliore aussi l' « efficacité » puisqu'il faut moins de temps pour réparer une erreur, et ceci d'autant plus qu'il en fait moins.

Pour être précis, le critère de gestion des erreurs est subdivisé en 3 sous critères. Le premier critère est celui de la **protection contre les erreurs** et concerne les moyens servant à détecter et prévenir les erreurs. Par exemple, un champ d'un formulaire obligatoire doit être spécifié.

Fournir la liste des valeurs possibles
Minimiser les saisies clavier
Les commandes à effet destructeur doivent être distinctement séparées des autres et placées en bas du menu. [Nog02]

Le second sous critère fait référence à la **qualité des messages d'erreur**, c'est-à-dire la pertinence, la facilité de lecture et l'exactitude de l'information donnée sur la nature des erreurs commises (syntaxe, format, etc.)

Utiliser le langage de l'utilisateur, de la tâche.
Eviter l'humour ou un vocabulaire réprobateur. [Nog02]

Le dernier sous critère est celui de la **correction des erreurs**. Il concerne les moyens disponibles pour l'utilisateur pour corriger immédiatement ses erreurs. L'utilisateur doit pouvoir accéder à la donnée erronée pour la modifier.

Mettre en évidence le ou les champs erronés. [Nog02]

4.2.6 Homogénéité / Cohérence

Ce critère se réfère à la façon dont les choix de conception de l'interface (codes, dénominations, formats, procédures, etc..) sont conservés pour des contextes identiques, et sont différents pour des contextes différents. Cela permet de rendre les données et les objets facilement identifiables, reconnaissables et utilisables. Standardiser l'interface et les procédures, conduit à la conception d'une interface homogène et stable, donc aussi plus prévisible. Cela permet un meilleur « apprentissage » de l'interface par l'utilisateur, ainsi qu'une plus grande « efficacité ». L'exemple du premier chapitre concernant le « look and feel » des applications Microsoft s'inscrit parfaitement dans ce critère, puisque la plupart de leurs applications sont conformes à leurs standards et conventions. Prenons

par exemple la structure d'un menu (file, edit, view, …, help) et les raccourcis clavier associés.

> *Les fenêtres doivent suivrent le même schéma d'agencement.*
> *La syntaxe des commandes doit être cohérente sur l'ensemble de l'interface. [Nog02]*

4.2.7 Signification des codes et dénominations

Ce critère concerne l'adéquation entre l'objet ou l'information affichée et son référent. C'est-à-dire que les codes utilisés, les items de menu, les libellés, facilitent l'encodage et la mémorisation. Par exemple, les raccourcis clavier doivent être facilement associables avec la procédure. De même des dénominations ou icônes non significatives pour l'utilisateur peuvent leur suggérer des opérations inadéquates et l'induire en erreur. Le respect de ce critère permet d'augmenter l' « apprentissage » et la « mémorisation » de l'interface, puisque l'utilisateur aura moins de difficultés à comprendre et à se rappeler la signification si celle-ci lui est déjà familière.

> *Les boutons et liens doivent refléter le contenu de l'interface de destination, de la page de destination. [Nog02]*

> *A successful Icon [Gal96]*
> *- looks different from all other icons*
> *- is obvious what it does or represents*
> *- is recognizable when no larger than 16 pixels square*
> *- looks as good in black and white as in color.*

4.2.8 Compatibilité

Un des critères le plus important est celui de la compatibilité, il concerne l'accord pouvant exister entre les caractéristiques des utilisateurs (mémoire, perceptions, habitudes, compétences, âge, attentes, etc.) et les caractéristiques des tâches, du système. On dira qu'une interface est compatible si et seulement si le (re)codage d'informations et de tâches du monde réel en données et actions du système est réduit. Il s'agit d'une cohérence avec l'environnement extérieur à l'application, et en particulier avec les caractéristiques de l'utilisateur. La compatibilité est vérifiée lorsque la logique d'utilisation du système correspond à la logique de l'utilisateur [Nog02]
Le respect de ce critère permet d'augmenter la « satisfaction » et l' « efficacité » de l'interface.

Un bon exemple pour illustrer ce critère est celui des couleurs. En effet, la couleur possède en soi une signification. Le rouge signifie une perte dans le domaine financier et signifie un interdit en signalisation routière. Alors que le vert signifie la sécurité, la végétation, l'exactitude. Ce n'est donc pas idéal d'afficher un message d'erreur en vert, car l'utilisateur associe le vert à l'exactitude et non à l'erreur. La compatibilité d'une interface dépend fort des caractéristiques des utilisateurs. En effet, la signification des couleurs varie selon les cultures. Le rouge par exemple signifie la mort en Afrique, la joie

en Asie et la pureté en Inde. Alors que le vert et l'orange ont une signification religieuse en Irlande.

> *Common meanings [Gal96]*
> *- to indicate that actions are necessary, use warm colors (Red, orange, yellow)*
> *- to provide status or background information, use cool colors (Green, blue, violet, purple)*
> *- Conform to human expectancies in the job, in the world at large*

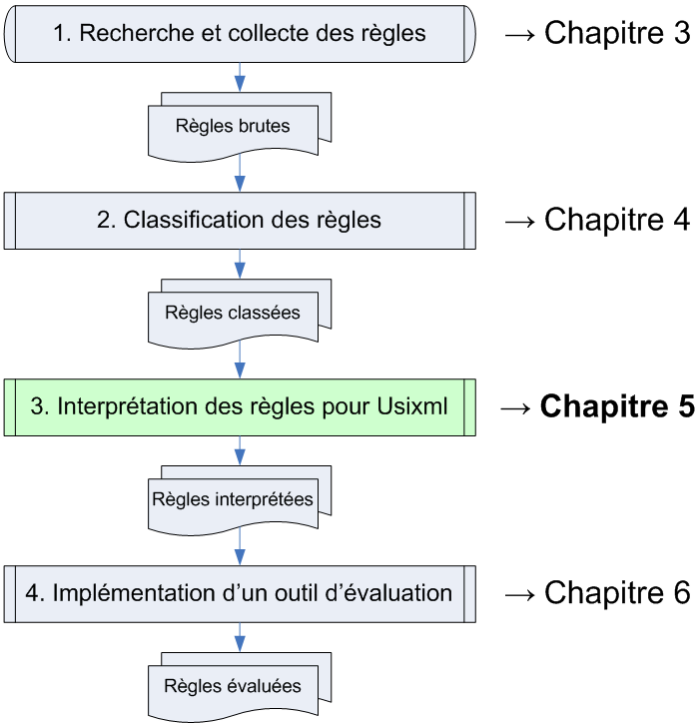
Maintenant que nous avons terminé la classification des règles, nous pouvons passer à l'étape suivante de notre méthodologie, c'est à dire l'interprétation des règles.

Chapitre 5 Interprétation des règles pour Usixml

Dans le chapitre précèdent nous avons classifié, regroupé, organisé les règles d'accessibilité et d'ergonomie. Nous pouvons donc passer à l'étape la plus importante de notre méthodologie, c'est-à-dire l'interprétation des règles dans un langage particulier.

Dans la première section, nous expliquerons que le choix d'un langage particulier conduit à se restreindre à un ensemble plus petit de règles. La section 5.2 présentera le langage particulier que nous avons choisi et sera suivie par la section 5.3 qui en présentera les limitations et restrictions liées au choix de ce langage. La section 5.4 sera consacrée à l'interprétation des règles que nous avons classifiées pour le langage choisi.

Avec l'interprétation des règles pour le langage particulier choisi, nous serons en mesure de les évaluer. C'est le sujet qui sera développé dans le chapitre suivant.



5.1 Restrictions à un langage particulier

Un langage informatique n'est pas aussi large qu'un langage naturel. C'est avec le langage naturel que sont définies les règles. Prenons par exemple la règle concernant les couleurs d'une interface qui dit à peu près la chose suivante :

> *Choisissez les couleurs qui rendront votre interface facile à lire pour les personnes victimes d'une déficience dans la perception des couleurs.*

Cette recommandation est totalement indépendante du langage utilisé pour représenter l'interface. Pour la tester, nous pouvons faire des tests réels, c'est-à-dire avec des utilisateurs victimes de ce type de handicap, et ainsi voir s'ils sont capables de lire l'interface. Cela dit, comme nous le présentions dans le premier chapitre, nous désirons pouvoir effectuer une validation automatique, plus rapide et plus économique. Pour cela, nous devons conceptualiser cette règle en des éléments, des informations calculables. Sur base d'étude sur les couleurs [Murch87], une interprétation de cette règle pour les interfaces peut être produite [Vand04].

> *La combinaison entre la couleur d'arrière plan et d'avant plan doit appartenir aux meilleures combinaisons de couleur et ne doit pas appartenir aux pires combinaisons de couleur proposées par Murch.*

Cette règle est déjà plus restrictive que la règle générale. La combinaison de couleur n'est pas un critère suffisant, mais certainement nécessaire, pour s'assurer que l'interface est facile à lire pour ce groupe de personnes.

Prenons maintenant un langage fictif particulier. Imaginons que celui-ci serait basé sur un langage composé de tags, de balises, d'attributs tout comme l'HTML, mais que ce langage ne contienne des indications de couleurs uniquement pour les textes. Nous serions en mesure de tester les couleurs d'arrière plan et d'avant plan des textes, mais uniquement de ceux là. C'est-à-dire que les tableaux, les cadres, les images, etc. ne seraient pas possibles à tester. Cela réduit la portée de notre recommandation et rend presque impossible de s'assurer que l'interface sera effectivement facile à lire.

Nous observons que tous les langages ont des limitations qui ne permettent pas de couvrir toute la portée des règles d'ergonomie générale [Att05]. Interpréter les recommandations restreint inévitablement la portée des règles générales, quel que soit le langage d'interface.

Dans la section suivante, nous présentons le langage particulier que nous avons choisi et qui, comme tout langage, ne permet pas de couvrir toute la portée des règles d'ergonomie.

5.2 Choix d'Usixml comme langage particulier

Comme le montre l'étude de R. Atterer et A. Schmidt [Att05], les langages basés sur des modèles offrent de nouvelles possibilités concernant la validation de règles d'ergonomie, puisqu'ils incluent des informations qui ne sont pas présentes dans la structure finale des interfaces utilisateurs. C'est dans l'optique d'étendre les études qui ont déjà été réalisées

que nous avons choisi le langage Usixml, puisque c'est un langage basé sur des modèles qui est actuellement développé par l'Université Catholique de Louvain.

5.2.1 Introduction à Usixml

Les développements des interfaces utilisateurs d'application interactive sont très difficiles à cause de la complexité et la diversité des environnements de développement existant et de la quantité de compétence de développement requis pour réaliser des interfaces utilisateur avec un haut degré d'utilisabilité. La difficulté est d'autant plus grande si l'on doit développer une même interface pour plusieurs contextes d'utilisation différents, c'est-à-dire, différentes catégories d'utilisateurs qui se distinguent par la langue, des préférences d'affichage, des handicaps, différentes plates-formes comme les téléphones portables, les ordinateurs de bureaux ou les kiosques interactifs. C'est la raison pour laquelle des langages de description d'interface utilisateur (UIDL) ont vu le jour, en s'orientant vers différents aspects des interfaces utilisateurs. Ces langages ont été introduits avec différents objectifs en tête : langage de spécification, format de communication, expression de la portabilité entre toolkits, support d'adaptation, support de l'indépendance de plate-forme, multimodal, multimédia, etc… USIXML (USer Interface eXtensible Markup Language) [Limb04] est un User Interface Description Language (UIDL) dont le but est de décrire l'interface utilisateur avec différents niveaux de détails et abstractions, dépendant du contexte d'utilisation. Il se targue de cumuler beaucoup des objectifs introduits par les autres UIDL comme : assurer la portabilité d'une plate-forme à une autre, garder les exigences de l'interface pour une définition abstraite qui reste stable au cours du temps, améliorer la réutilisabilité, produire une conception d'interface pour plusieurs plates-formes et outils, supporter l'extensibilité et l'adaptabilité ou encore utiliser une description d'interface permettant une génération automatique du code. Mais aussi, supporter l'intégration de tout modèle utilisé dans le processus de développement des UI, exprimer explicitement les correspondances entre les modèles, etc.

Dans notre analyse, nous nous intéressons surtout à l'aspect multi plates-formes d'Usixml. Usixml est un langage de description compatible avec XML. Ce langage décrit les interfaces utilisateurs pour des contextes d'utilisations multiples telles que les interfaces utilisateurs de type caractères (CUI : Character User Interface), les interfaces utilisateurs de type graphique (GUI : Graphical User Interface), les interfaces utilisateurs de type auditif, et les interfaces utilisateurs de type multimodal. Grâce à Usixml, les personnes qui ne sont pas des développeurs peuvent décrire l'interface utilisateur de toutes nouvelles applications interactives en spécifiant la description dans Usixml, sans avoir les compétences de programmation habituellement nécessaires pour les langages à balises (markup language) tel que HTML ou les langages de programmations comme Java ou C++.

5.2.2 Structure d'Usixml

Afin de bien comprendre comment Usixml fonctionne dans la réalisation d'une interface utilisateur, il est intéressant de savoir qu'Usixml se base sur le framework Cameleon [Cam05].

Usixml est structuré en accord avec les 4 niveaux de base d'abstraction défini par le Framework Cameleon de référence (voir Figure 13) dont le but est d'exprimer le cycle de vie du développement d'interface utilisateur pour les applications interactives sensibles au contexte.

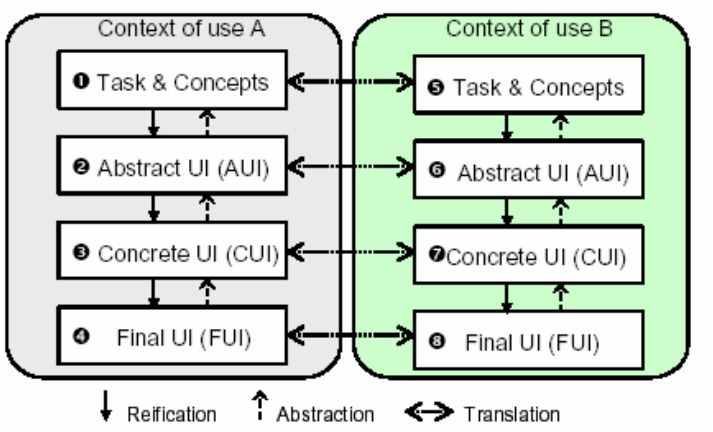


Figure 13 - The Cameleon Reference Framework

En partant du bas, nous avons le *« Final User Interface »* (FUI), c'est-à-dire l'interface utilisateur final tournant sur une plate-forme particulière. Une interface utilisateur final est généralement le code d'une interface qui est exprimé dans un langage particulier de rendu, tel que HTML, Java, etc.
Au-dessus, au troisième niveau, nous avons le *« Concrete User Interface »* (CUI), c'est une abstraction du FUI, indépendant de la plate-forme d'exécution. Ce niveau est composé d'objets d'interaction concrets, qui définissent la mise en page et la navigation de l'interface.
Au deuxième niveau, nous avons l' *« Abstract User Interface »* (AUI). L'AUI abstrait le CUI en une définition qui est indépendante de toutes modalités d'interaction. C'est-à-dire indépendante des interactions graphiques ou vocales, des synthétiseurs de la parole, de la reconnaissance de la voix ou encore des interactions à réalité augmentée. L'AUI est aussi considéré comme une expression canonique du rendu du modèle des tâches et contextes.
Tout en haut du framework, nous avons le niveau des *« Tâches et Concepts »* (T&C). Il décrit les différentes tâches à effectuer et les différents concepts du domaine, ainsi que la manière dont ils doivent être exécutés.

En plus des 4 niveaux, il existe 3 relations de base :

1. L' « Abstraction » qui va d'un niveau inférieur vers un niveau supérieur et qui transforme toute spécification en une abstraction de plus haut niveau.
2. La « Réification » qui agit de manière inverse, d'un niveau supérieur vers un niveau inférieur, qui réduit l'abstraction des spécifications. Elle les concrétise.
3. La « Translation » qui est une transformation de l'interface d'un contexte d'utilisation à un autre.

Afin d'illustre ces relations, nous pouvons voir dans la Figure 14 , comme la page de recherche de Google peut être abstraite, ou comme un modèle de tâche peut se transformer en une réalisation concrète.

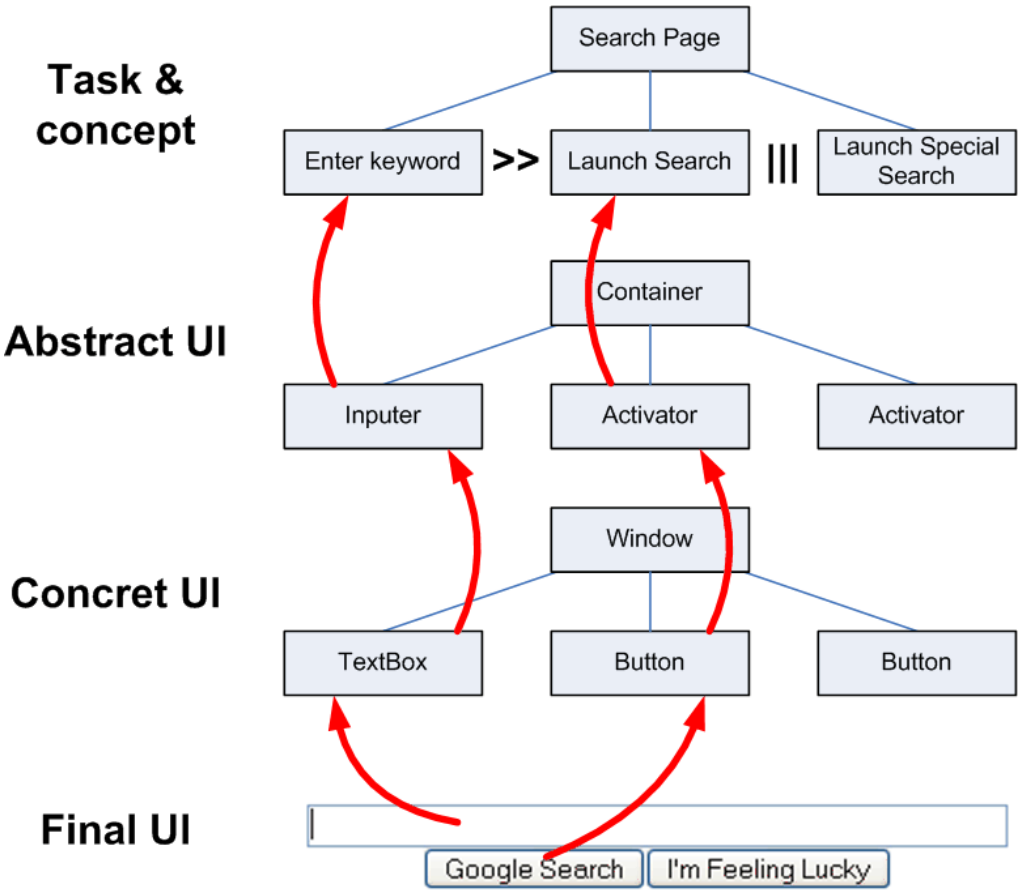


Figure 14 - Abstraction d'un page de recherche en un modèle de tâches.

La structure d'USIXML se base sur ce framework. Il est composé de plusieurs modèles :

1. **Modèle de tâche** (Task Model): Correspond au premier niveau du framework et décrit les différentes tâches. C'est une version étendue du ConcurTaskTree (CTT) [Pate99] qui est utilisée pour représenter les tâches utilisateurs et leur ordre logique et temporel.

2. Modèle du domaine (Domain Model) : Inclus également dans le premier niveau du framework, il décrit les concepts du monde réel et leurs interactions telles que comprises par l'utilisateur. Ce modèle à la forme d'un diagramme de classe UML. Les concepts contenus dans le modèle du domaine sont à un certain moment utilisé par l'utilisation, c'est-à-dire que les valeurs sont rendues à travers l'interface, que les méthodes attachées aux classes d'objets sont déclenchées par des événements de l'utilisateur.

3. Modèle de contexte (Context Model): Ce modèle décrit toutes les entités qui peuvent influencer l'exécution des tâches interactives de l'utilisateur. Ce modèle est constitué de 3 modèles :

- Le modèle utilisateur (User Model): Ce modèle décompose récursivement la population en stéréotypes, en profil. Chaque stéréotype partage une même série d'attributs et de valeurs associées.
- Le modèle de la plate-forme (Platform Model): Ce modèle contient les attributs pertinents de chaque plate-forme qui peuvent influencer significativement le contexte.
- Le modèle d'environnement (Environment Model): ce modèle décrit toutes les propriétés de l'environnement physique du lieu où l'interface sera utilisée.

4. Modèle AUI (AUI Model): Comme son nom l'indique, il correspond au second niveau du framework Cameleon. Il représente l'expression canonique de rendu et manipulation des concepts du modèle et des fonctions de manière indépendante de toute modalité et plate-forme de calcul.

5. Modèle CUI (CUI Model): Ce modèle autorise la spécification de l'apparence et du comportement de l'interface avec des éléments qui peuvent être perçus par l'utilisateur. Ce modèle est dépendant des modalités, pour l'instant Usixml supporte deux modalités : graphique et auditive. Par contre ce modèle est indépendant de la plate-forme, les objets d'interaction concrets (CIO) sont une abstraction d'un ensemble de widget trouvés dans les toolkits graphiques courants (Java AWT, Swing, HTML 4.1, …), comme par exemple un bouton, un check box, une liste, etc.

6. Modèle de correspondance inter modèle (Inter-Model Mapping Model): Ce modèle est un peu particulier. Il contient un ensemble de relations prédéfinies permettant de correspondre des éléments sémantiquement reliés entre les modèles. Les correspondances entre les différents modèles sont de différents types :
   - *Manipulates* : correspondance d'une tâche à un concept du domaine (une classe, un attribut, une opération, …)
   - *Is Rendered By* : correspondance d'un concept du domaine en un élément de présentation soit que le concept du domaine est sujet à recevoir des infos de l'utilisateur soit qu'il est seulement présenté à l'utilisateur.
   - *Is Executed In* : correspondance d'une tâche dans un élément des modèles AUI ou CUI.
   - *Is Abstracted Into* et *Is Reified Into* : correspondance entre des éléments des modèles AUI et CUI.
   - *Has Context :* correspondance entre un élément de n'importe quel domaine et un ou plusieurs contextes d'utilisation.
   - *Corresponds To :* correspondance d'une relation temporelle de tâche avec une relation de navigation définie dans le modèle AUI ou le modèle CUI.

Notons que nous n'avons pas de modèle correspondant au dernier niveau du framework Cameleon. En effet, l'un des avantages d'Usixml, c'est qu'il reste indépendant de la plate-forme. Si nous voulions ajouter un niveau d'interface utilisateur final, nous devrions choisir un langage et Usixml ne serait plus totalement indépendant. Cela dit, nous pourrions développer des modèles pour le niveau d'interface utilisateur final pour chaque langage de programmation et les intégrer à Usixml.

5.2.3 Information contenue dans chaque modèle

Il est impossible de présenter toutes les informations contenues dans le langage Usixml, Nous allons cependant balayer brièvement chaque modèle et y repérer les informations qui sont susceptibles de nous aider dans notre étude de validation ergonomique.

5.2.3.1 Le modèle de tâches

Ce modèle est composé de tâches et de relations entre elles. La classe tâches (task) possède plusieurs attributs intéressants :

Le type d'interaction au sens de Paterno qui est une information intéressante pour savoir dans quelle mesure l'utilisateur contrôle le système. C'est-à-dire que si l'on observe que beaucoup de tâches s'effectuent sans que l'utilisateur ait réalisé une action déclenchant cette tâche, l'interface n'est pas conforme au critère d'ergonomie : « Contrôle explicite ».

Les attributs frequency et importance sont particulièrement intéressants puisque dans beaucoup de règles d'ergonomie, il est souvent question de classifier les commandes selon leur importance ou leur fréquence. Si nous n'avons pas cette information, nous sommes obligés de demander à l'utilisateur d'indiquer l'importance de chaque commande présente dans un menu, dans une fenêtre, etc. Cela revient à demander au superviseur de l'évaluation de faire la correction lui-même.

Deux autres attributs structurationLevel et complexityLevel peuvent nous aider dans la validation de règles, mais sont moins évidentes.

Les attributs userAction et taskItem sont par contre très intéressants, ils nous permettent de savoir si la tâche requière une action de l'utilisateur et surtout de quel type est cette action. start/go, stop/exit, etc., ainsi que le type d'objet ou sujet de l'action : operation, container, collection ou element.

Concernant les relations entre les tâches, les relations unaires sont particulièrement intéressantes dans le sens où elles nous permettent d'indiquer lorsqu'une tâche est optionnelle, itérative, ainsi que le nombre de fois qu'elle s'itère.

5.2.3.2 Le modèle du domaine

Ce modèle est composé d'objets, de classe du domaine et de relations entre les classes du domaine. Une classe du domaine possède un ou plusieurs attributs et méthodes.

Chaque attribut est défini par un nom, un type de donnée, une cardinalité minimum et maximum, ainsi qu'une caractérisation du domaine. Un ensemble de valeurs énumérées est également possible. Cette information peut être intéressante pour tester des règles d'ergonomie concernant le nombre de valeurs énumérées, leur syntaxe, etc.

Un objet est une instanciation d'une classe. Chaque objet contient des instances de ses attributs, c'est-à-dire une valeur pour cet attribut. Ces valeurs sont utilisées pour indiquer les valeurs de départ de certains champs éditables. Or il existe des règles d'ergonomie qui

demandent que chaque champ éditable ait une valeur par défaut (au lancement de l'interface) :

> *WCAG 1.0 - 10.4 Until user agents handle empty controls correctly, include default, place-holding characters in edit boxes and text areas.*

5.2.3.3 Le modèle abstrait

Ce modèle nous rapproche un petit peu plus de la version finale que l'utilisateur utilisera. En effet, ce modèle fait apparaître les AIO (Abstract Interaction Objet), c'est-à-dire les objets avec lequel l'utilisateur interagira mais de manière encore abstraite.

Chaque AIO est d'un type particulier, soit un composant individuel abstrait qui prend en charge au moins une « fonction » de l'interface utilisateur, soit un conteneur abstrait contenant d'autres AIO, avec une indication d'un ordre entre ces AIO , ces autres AIO peuvent être d'autres conteneurs abstraits ou des composants individuels abstraits. C'est informations sont importantes pour connaître la structure de l'interface naissante. On peut déjà voir si certains conteneurs sont surchargés par rapport à d'autres, afin de produire une interface équilibrée. Beaucoup de règles pourront s'appliquer de manière similaire au modèle abstrait et au modèle concret.

Un conteneur abstrait est un type d'AIO qui supporte l'exécution d'un ensemble de tâches logiquement ou sémantiquement connectées. Cette information est également intéressante, puisque nous savons qu'une règle d'ergonomie nous suggère de présenter les tâches logiquement ou sémantiquement dans une même fenêtre. Si nous savons cela, nous pouvons vérifier qu'un conteneur abstrait soit bien traduit en une et une seule fenêtre et non plusieurs.

Un composant individuel abstrait assume différentes fonctions (au moins une) décrites par des facettes. Chaque facette décrit une fonction particulière qu'un AIO peut assumer. Il y a 4 types de facettes principales :

1. Les facettes d'input qui décrivent le type d'input qui peut être accepté par l'AIO
2. Les facettes d'output qui décrivent quelle donnée peut être présentée à l'utilisateur par l'AIO
3. Les facettes de navigation qui décrivent les transitions possibles entre containers qu'un AIO particulier peut permettre
4. Les facettes de contrôle qui décrivent les méthodes possibles des fonctionnalités noyaux qui peuvent être déclenchées par un widget particulier.

Une facette possède, en outre, deux attributs : actionType et actionItem. L'actionType permet de spécifier le type de l'action que l'AIO permet d'exécuter. Les valeurs permises sont les mêmes que l'attribut userAction du modèle de tâche. L'actionItem permet de caractériser l'élément qui est manipulé par le composant abstrait. Cela peut être un objet, un attribut, une méthode, un ensemble d'objet, un ensemble d'attribut.

Pour les facettes d'input, nous avons les informations concernant le type de donnée et sa cardinalité. Particulièrement intéressant pour tester la présence d'une aide après le champ à éditer, indiquant le type de donnée à introduire par exemple.

Pour les facettes de contrôle, nous avons, en plus, l'événement, l'action et la priorité afin d'ordonner les actions dans le cas où le composant individuel abstrait a plusieurs facettes de contrôle. Les éléments de contrôle sont particulièrement intéressants, puisqu'il est bon de savoir si l'utilisateur reçoit un feed-back pour les événements qu'il produit.

Pour les relations entres les AIOs, il en existe de 3 types : spatioTemporal, mutualEmphasis et dialogControl. La relation de type spatioTemporal, par exemple, décrit une contrainte abstraite entre les AIOs en termes de temps et d'espace indépendamment de toute modalité d'interaction. Dans le cas d'une contrainte temporelle, seule une dimension est indiquée (selon les relations d'Allen), et dans le cas d'une contrainte spatiale, les deux dimensions (axe X et axe Y) sont indiquées. Les informations concernant la mise en page de l'interface sont intéressantes pour tester des règles de présentation telle que la symétrie, l'équilibre, etc.

5.2.3.4 Le modèle concret

C'est le modèle que nous allons utiliser le plus. Il est le dernier modèle avant transformation vers la version finale. Ce modèle n'est plus totalement indépendant des modalités.

- **Les comportements des objets**

Comme les autres modèles, ce modèle est composé de CIO (Concret Interaction Object) et de relations entre ces objets. Chaque objet a également une partie d'information concernant son comportement. Plus exactement, on peut indiquer quels sont les événements qui indiquent l'initiation et la terminaison d'un CIO, ainsi que ses comportements. L'attribut eventType permet de définir le type de l'événement. Le type d'événement peut être l'un des suivants : movePointer, pointerOver, moveOutPointer, click, doubleClick, depress, release, dragOver, dragDrop, hasFocus, lostFocus, resize, change, move, spinUp, spinDown. Beaucoup de règles d'ergonomie et d'accessibilité font référence au comportement des objets. A titre d'exemple, une des règles d'accessibilité est d'indiquer à tout moment quel est l'objet qui a le focus. Ce type d'information est donc très utile pour la validation.

> *IBM Software – Object Information*
> *2.1 Provide a visual focus indicator that moves among interactive objects as the input focus changes. This focus indicator must be programmatically exposed to assistive technology.*

Un événement dépend aussi de l'outil pour déclencher l'événement, l'attribut device sert à renseigner quel est l'outil avec lequel l'événement a été déclenché.

Un objet peut avoir plusieurs comportements. Un comportement est composé d'une action, de conditions (pré & post) dans lesquelles il exécute l'action et d'un événement. Il

est possible de spécifier les pré- et post-conditions attachées à une action. Une condition est exprimée comme étant une combinaison par des opérateurs logiques de termes de règles qui sont composés de tout fragment du model Usixml. Pour une explication complète du mécanisme, nous vous renvoyons à [Limb04].

Une action est un processus déclenché par l'exécution d'un événement sur un CIO. Une action peut être un appel de méthode, un système de transformation ou une transition entre deux conteneurs. Ces informations sont particulièrement intéressantes pour tester les règles concernant la navigation, le feed-back utilisateur, etc.

- **Les relations entre les objets**

Une relation entre les objets peut être de 3 types différents au niveau du modèle concret : les relations audio, les contrôles de dialogue et les relations graphiques.

Les relations audio sont utilisées pour les objets audio et sont de deux types : transition et adjacence. Pour les relations d'adjacence, on peut exprimer le délai en secondes entres les deux éléments audio. Pour les relations de transition, on peut définir le type de la transition : open, mute, reduce volume, restore volume ainsi qu'une spécification de l'effet auditif de la transition, par exemple : fade-out, fade-in.

Les relations graphiques se divisent encore en plusieurs types de relations, on y fait la différence entre l'adjacence, l'emphase, l'alignement et la transition. La transition graphique est une relation entre un ou plusieurs conteneurs graphiques. Cela permet de spécifier une structure de navigation entre les différents conteneurs que contient le modèle concret. On indique pour chaque relation, le type qui peut être : open, close, minimize, maximize, suspend/resume. Une indication concernant l'effet de transition peut également être renseignée. Ces informations peuvent nous être très utiles pour toutes les règles d'ergonomie faisant référence à la navigation, au parcours de l'utilisateur dans l'interface.

Les relations d'alignement permettent de spécifier des contraintes d'alignement entre deux composants graphiques individuels (isVertical, isHorizontal, isRightCenterLeft, isUpDown). Ces indications permettent d'aligner les objets entre eux selon l'axe vertical et l'axe horizontal. Ce sont, à notre sens, des informations très utiles pour connaître la mise en page de l'interface et valider des règles tel que :

> *Differentiate captions from single data fields:*
> - *place the caption to left to data field*
> - *Align the caption with the control's data.*
> - *Alternatively, place the caption above the data field*
> - *Align justifier upper-left to the data field* *[Gal96]*

Les relations d'emphase graphique permettent de spécifier que deux ou plusieurs composants graphiques individuels doivent être différenciés d'une certaine manière (par des couleurs différentes, par exemple). Il existe beaucoup de règles concernant les techniques d'emphase et surtout comment les utiliser.

Limiter le nombre total de moyens de mise en évidence utilisés sur l'interface (5 au maximum) [Nog02]

- **Les CIO (Concret Interaction Object)**

Les objets peuvent être graphiques, audio ou représenter déjà un composant final. Dans tous les cas, un objet possède un attribut icon et defaultIcon permettant d'associer une icône (dépendant du contexte ou non) à l'objet. Les attributs content et defaultContent sont les attributs les plus importants puisqu'ils donnent le contenu de l'objet. Les deux derniers attributs defaultHelp et help permettent de définir une aide pour l'objet (dépendante du contexte ou non). La présence d'icône permet d'aider les personnes à utiliser les interfaces, mais trop d'icônes nuît à la qualité ergonomique de l'interface également.

Limiter le nombre d'icônes : 12 au mieux, 20 au maximum. [Nog02]

- **Les objets audio**

Les objets audio sont soit des conteneurs soit des composants audio individuels de type input ou output. Un composant audio individuel de type output peut être de la musique, de la voix, ou une icône sonore. Parmi les attributs, l'attribut isMuted indique si l'output sonore est rendu muet ou non à l'origine. Il est particulièrement intéressant pour tester si l'objet sonore est joué directement ou si l'utilisateur doit effectuer une action pour le rendre audible. Ceci afin de répondre au critère de contrôle explicite de l'utilisateur.

- **Les objets graphiques**

Les objets graphiques sont les plus nombreux. Parmi les attributs communs à tous les objets graphiques, les attributs isVisible, isEnabled, fgColor, bgColor, toolTipDefaultContent, toolTipContent sont particulièrement intéressants pour valider les règles d'ergonomie concernant les couleurs, les alternatives textuelles, etc. D'autres attributs comme borderWith, borderType, defaultBorderTitle, borderTitle, borderTitleAlign et borderColor permettent de définir les bordures d'un compostant graphique, et nous aidera à valider des recommandations comme la suivante :

Grouping Using Borders :
- *Incorporate line borders for :*
  - *Focusing attention on groupings or related information.*
  - *Guiding the eye through a screen*
- *Do not exceed three line thickness or two line styles on a screen, however.*
  - *Use a standard hierarchy for fine presentation*
- *Create lines consistent in height and length.*
- *Use rules and borders sparingly*

- **Les conteneurs graphiques**

Les conteneurs graphiques sont les objets graphiques qui contiennent une collection d'autres objets graphiques. Ils peuvent être de différents types, dont le plus évident est celui de windows. Lorsque le conteneur graphique est une fenêtre, nous pouvons indiquer

(en pixel) les marges à gauche et supérieures par les attributs windowLeftMargin et windowTopMargin. De même nous pouvons indiquer si la fenêtre est redimensionnable ou non grâce à l'attribut isResizable. Un autre type de conteneur graphique est la boite, ou « box ». C'est l'un des plus employé, avec les tables, pour définir le design d'une interface. D'ailleurs l'attribut isBalanced qui indique que tous les composants graphiques individuels sont topologiquement balancés dans la boite, est particulièrement intéressant pour valider la recommandation suivante, issu du livre de Galitz [Gal96].

> *Balance :*
> - *Create screen balance by providing an equal weight of screen elements, left and right, top and bottom.*

L'autre type de conteneur graphique fortement utilisé pour définir le design d'une interface est le tableau (ou table) composé de cellules. Les cellules sont des conteneurs en eux-mêmes mais doivent faire partie d'un tableau. Une cellule a trois attributs afin de savoir à quel endroit il se positionne dans le tableau. xIndex, yIndex et zIndex désignent la ligne, la colonne et la couche

Les boîtes de dialogues tabulées sont des groupes de boîtes de dialogues où chaque boîte de dialogue est accessible par un mécanisme de tabulation. Pour chaque tab d'une boîte de dialogue tabulée, il y a un élément tabulé. On renseigne, pour chaque élément tabulé, un index, ceci afin de les ordonner entre eux.

- **Les composants graphiques individuels**

Les composants graphiques individuels sont contenus dans des conteneurs graphiques. Les attributs glueVertical et glueHorizontal qui peuvent respectivement prendre les valeurs top, middle, bottom et left, middle, right permettent de spécifier la relation de mise en forme avec son conteneur graphique. Les attributs defaultMnemonic et Mnemonic des composants graphiques individuels contiennent les mnémoniques par défaut pour contrôler le composant. C'est évidement une information importante puisqu'il existe des règles d'accessibilité les concernant. Les composants peuvent être de différents types. Des composants de texte, d'image, de vidéo, des boutons, ou d'autres widgets.

- **Les composants textes**

Les composants textes ont beaucoup d'attributs : textFont, textSize, textColor, isBold, isItalic, isUnderline, isStrikeThrough, isSubScript, isSuperScript permettent de formater le contenu textuel, d'y mettre de l'emphase, etc. Il existe des montagnes de recommandations pour la typographie comme la suivante, issue du livre de J-F Nogier [Nog02]

> *Ne pas utiliser plus de 4 polices de caractères.*
> *Utiliser une police droite.*

Les attributs defaultHyperLinkTarget, hyperLinkTarget, linkVisitedColor, activeLinkColor sont des attributs pour spécifier les liens hypertextes, les couleurs des liens visités et actifs.
De plus, les attributs textMargin, isEditable, wordWrapped, forceWordWrapped, maxLength, numberOfColumns, numberOfLines, textVerticalAlign et textHorizontalAlign donnent des

informations sur la présentation des textes. Cela nous permettra de tester des recommandations comme les suivantes issues de [Gal96]

Text Presentation :
- *Include no more than 40-60 characters on each line.*
  - *A double column of 30-35 characters separated by 5 spaces is also acceptable*
- *Do not right-justify*

- **Les composants vidéo**

Concernant les composants vidéo, ils possèdent les attributs suivants : alternateImage, autoplay, loop, subtitle et subtitleContent. Ces informations seront utilisées pour les recommandations d'accessibilité de type « Fournir des alternatives ».

- **Les composants image**

Un composant image peut avoir zéro, une ou plusieurs zones d'images. Les attributs les plus intéressants d'une zone image sont hyperLinkTarget et defaultHyperLinkTarget qui désignent les liens hypertextes de cette région. Ces informations sont particulièrement importantes pour les recommandations d'accessibilité suivantes :

WCAG 1.0 - 1.2 Provide redundant text links for each active region of a server-side image map.

WCAG 1.0 - 1.5 Until user agents render text equivalents for client-side image map links, provide redundant text links for each active region of a client-side image map.

- **Les autres composants graphiques individuels**

Il y a beaucoup d'autres types de composants graphiques individuels, dont les boutons qui sont de 3 types : button, radioButton et toggleButton, les checkbox, les comboBox, les spin, les slider, les cursor , les progressionBar et d'autres widgets qui permettent de sélectionner des informations particulières comme l'heure, la date, une couleur, etc. (datePicker, hourPicker, colorPicker, filePicker). Certains de ces widgets doivent respecter certaines recommandations d'ergonomie. En voici un exemple :

Checkbox – Size:
- *Show a minimum of one choice, a maximum of eight in a grouping.*

RadioButton – Structure:
- *A columnar orientation is preferred manner of presentation*
- *Left align the buttons and choice description.*
- *If vertical space on the screen is limited, orient the buttons horizontally*
- *Enclose the buttons in a border to visually strengthen the relationship they possess.*

- **Les menus**

Les menus sont aussi inclus dans le langage Usixml, Un menu est un composant graphique individuel composé d'éléments de menu (menuItem), pouvant eux-mêmes contenir des menus. Un menu permet donc la sélection d'un ou plusieurs éléments dans une liste d'éléments de menu. Un contenu, une icône et un type sont attribués à un élément de menu. Le type est indiqué par l'attribut « type » qui peut prendre les valeurs : commande, dialogue, sub-menu, toggle, radio, separator. De plus, des raccourcis sont attribués aux éléments de menu. Les attributs keyboardShorcut et defaultKeyboardShorcut permettent de définir les raccourcis clavier. L'information sur les raccourcis clavier est particulièrement intéressante pour nous permettre d'évaluer la recommandation d'accessibilité suivante :

> *WCAG 2.0 - Guideline 2.1 Make all functionality operable via a keyboard interface.*

5.3 Limitations et réductions dues à Usixml

Maintenant que nous avons observé la structure d'Usixml et des informations que ce langage contient, nous allons voir quelles sont les limitations et réductions propres au langage Usixml. Comme nous l'avons montré dans la première section de ce chapitre, chaque langage a des limitations, des particularités qui restreignent les possibilités de valider les règles ergonomiques dans toute leur couverture. Usixml n'est certainement pas un langage parfait et complet. Il ne couvre pas toutes les possibilités qu'offre l'ensemble des langages d'interface réunis. Usixml se concentre sur certains aspects, et ne permet évidement pas de tout couvrir.

Nous avons trouvé quatre grandes limitations et réductions inhérent à l'usage d'Usixml comme langage pour la validation automatique de règles ergonomiques. Nous les présentons dans les sous-sections suivantes.

5.3.1 Limitation liée au manque de FUI.

L'une des principales limitations d'Usixml vient de son indépendance vis-à-vis du rendu final de l'interface. Comme nous l'avons dit, il s'agit d'un avantage du langage, puisqu'il lui permet d'être beaucoup plus général. Il est indépendant du langage utilisé pour coder l'interface finale qui sera interprétée (navigateur web) ou compilée. Par contre, cela restreint notre ensemble de règles possibles à évaluer. En effet, certaines règles concernent le rendu final de manière très précise. Prenons, par exemple, quelques règles concernant les lignes de séparation dans un menu. Il est dit dans le livre de Galitz, The Essential Guide to User Interface Design (page 262), les recommandations suivantes :

> *Line Separators :*
> - *Separate vertically arrayed groupings with subtle solid lines.*
> - *Separate vertically arrayed sub-groupings with subtle dotted or dashed lines.*
> - *Left-justify the lines under the first letter of the columnized item descriptions.*
> - *Right-justify the lines under the last character of the longest item description.*

Ces recommandations ont été écrites pour des interfaces utilisateurs à caractères. D'ailleurs la seconde recommandation n'est pratiquement plus utilisée. Mais la première et la troisième restent souvent appliquées, comme en témoignent les menus des applications Office. Mais rares sont les logiciels qui appliquent encore la dernière des recommandations. Dans Usixml, il n'est pas possible d'évaluer si les trois dernières recommandations sont vérifiées. La troisième est liée à l'état actuel de développement d'Usixml, puisqu'il n'est pas possible d'indiquer le type de ligne de séparation (continue, pointillée, etc.). Par exemple pour obtenir le résultat à la Figure 15, nous écrivons le code suivant :

```
<menubar id="menubar_2" name="menubar_2" isVisible="false" isEnabled="true">
    <menu id="menu_4" name="menu_View" isVisible="true" isEnabled="true"
     popUpMenu="true" toolBarMenu="false">
      <menuItem id="menuitem_1" name="toolBox" isVisible="true" I
        isEnabled="true" type="toggle" keyboardShortcut="ctrl+T"/>
      <menuItem id="menuitem_2" name="colorBox" isVisible="true" I
        isEnabled="true" type="toggle" keyboardShortcut="ctrl+L"/>
      <menuItem id="menuitem_3" name="statusBar" isVisible="true" I
        isEnabled="true" type="toggle" />
      <menuItem id="menuitem_4" name="textToolBar" isVisible="true" I
        isEnabled="false" type="toggle" />

     <menuItem id="menuitem_5" name="separator" isVisible="true" I
        isEnabled="false" type="separator" />

      <menuItem id="menuitem_6" name="Zoom" isVisible="true" I
       isEnabled="true" type="radio" />
      <menuItem id="menuitem_7" name="ViewBitMap" isVisible="true" I
        isEnabled="true" type="radio" keyboardShortcut="ctrl+F"/>/>
   </menu>
    <menu id="menu_4" name="menu_Image" isVisible="true" isEnabled="true"
     popUpMenu="true" toolBarMenu="false"/>
    <menu id="menu_4" name="menu_Colors" isVisible="true" isEnabled="true"
     popUpMenu="true" toolBarMenu="false"/>
</menubar>
```

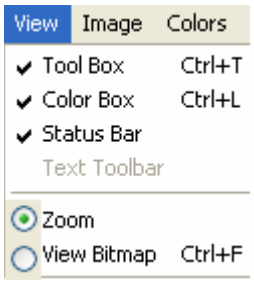


Figure 15 - Illustration d'une ligne de séparation dans un menu

Prenons un autre exemple pour illustrer la difficulté d'évaluation de recommandations spécifiques au rendu visuel de l'interface finale. Une des recommandations d'accessibilité, éditée par la section 508 et qui s'appliquent aux interfaces web, est la suivante :

> *Documents shall be organized so they are readable without requiring an associated style sheet.*

Il est sûr que ce type de recommandation est beaucoup trop lié à un certain langage de rendu, à savoir l'HTML et CSS.

5.3.2 Limitation liée à la structure en modèle d'Usixml

Comme nous l'avons vu, Usixml est structuré en couches, avec pour chaque couche un modèle. Le langage Usixml est particulièrement intéressant parce qu'il permet de générer automatiquement un modèle à partir d'un autre modèle. Que ce soit de l'abstrait vers le concret ou l'inverse, du concret vers l'abstrait. Cette spécification du langage est particulièrement intéressante dans le cadre de la reverse engineering. Pour passer d'un modèle à l'autre, on fait appel au modèle de transformation qui contient les règles de transformation.
Même si Usixml n'intègre pas de FUI, il n'empêche que la génération automatique de code HTML, Java, Qtk, etc. à partir du modèle concret est possible [Den05] [Oca05]. Le logiciel GrafiXML [Gra05], possède d'ailleurs des plug-ins d'exportation vers ces langages.

Les règles de transformation transforment les informations contenues dans un modèle en des informations d'un autre domaine. Dans toute transformation, il y a changement, il y a des choix. Certaines informations contenues dans un modèle peuvent ne pas être transformées en informations du nouveau modèle

Prenons quelques exemples de transformation du modèle abstrait vers le modèle concret.

Dans le modèle abstrait, nous avons des informations concernant les conteneurs abstraits. Comme nous l'avons décrit au point 5.2.3.3, un conteneur abstrait contient les tâches qui sont logiquement, sémantiquement connectées. Imaginons maintenant que lors des transformations vers le modèle concret, les règles de transformations (ou le concepteur) transforment notre conteneur abstrait en deux connecteurs concrets de type fenêtres. Il y a peut-être une autre raison à cette séparation, comme par exemple la taille maximale d'une fenêtre pour une visualisation sur PDA.

Le modèle abstrait contient des abstractIndividualComponent qui possèdent des facettes d'input, d'output, de navigation ou de contrôle. Si nous n'avions que le modèle abstrait et que l'on nous demandait de vérifier que l'interface soit équilibrée, nous validerions le schéma de la Figure 16 puisque chacun des conteneurs possède un même nombre d'objets abstraits de même type.

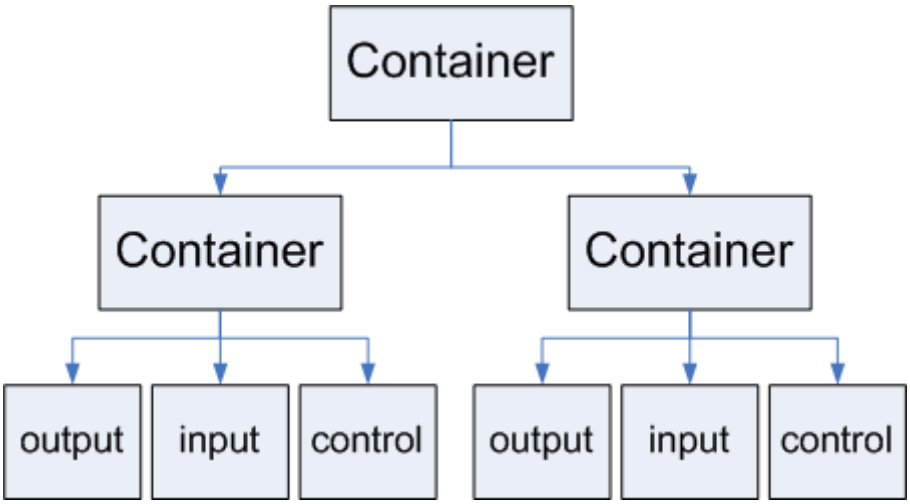


Figure 16 - L'équilibre est vérifié au niveau abstrait

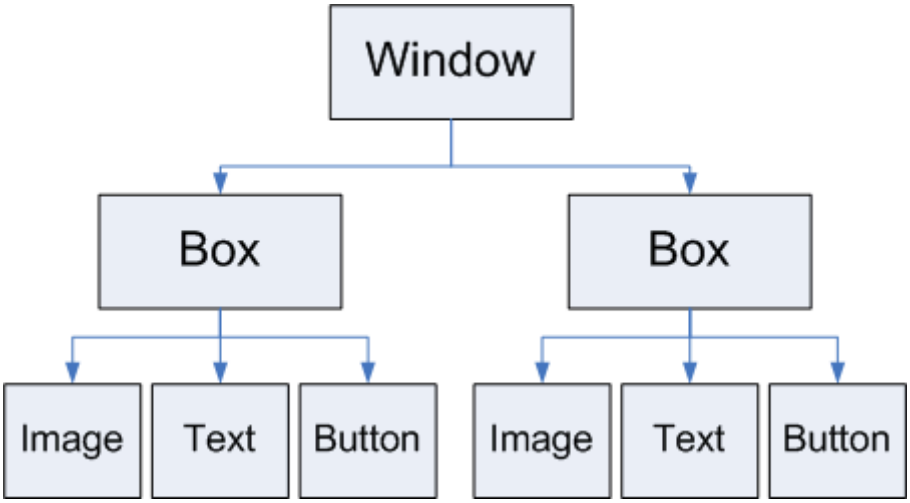


Figure 17 - L'équilibre n'est peut-être pas respecté, cela dépend des attributs des éléments.

Par contre, si nous avions un même schéma dans le modèle concret (voir la Figure 17) qui pourrait être issu du modèle abstrait, nous devrions prendre en compte la dimension des images et la longueur, la dimension des textes. En effet, on aura beau avoir un équilibre des composants, ce n'est pas pour cela que nous aurons le même équilibre dans le modèle concret, ou dans le rendu final de l'interface utilisateur, comme l'illustre la Figure 18.

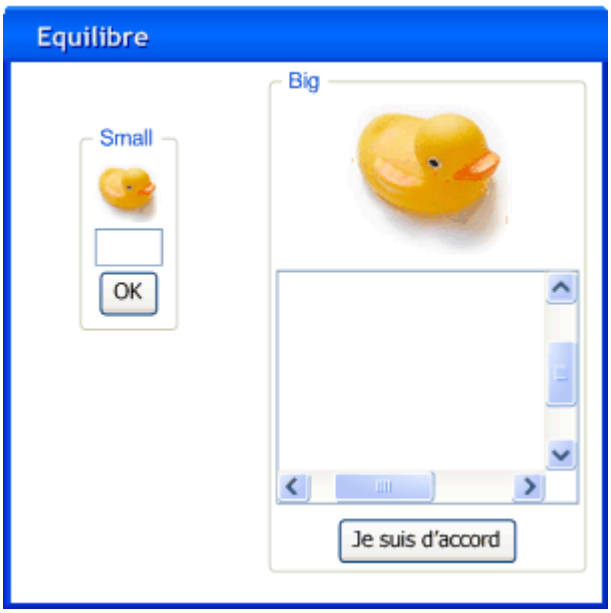


Figure 18 - L'équilibre n'est plus respecté dans le rendu de l'interface finale

Dans nos deux exemples, nous aurions certifié valide le modèle abstrait pour les deux règles illustrées, mais le modèle concret issu du modèle abstrait ne pourrait plus être certifié valide pour ces mêmes recommandations.

L'exemple est encore plus frappant dans les transformations entre le modèle concret et l'interface utilisateur finale. Comme nous l'avons déjà vu au point précédent concernant la limitation due au manque de FUI, nous ne pouvons pas contrôler le rendu final de l'interface utilisateur, mais nous ne pouvons pas non plus assurer que les informations contenues dans le modèle concret serons bien retransmises dans l'interface utilisateur. Prenons par exemple le langage XUL, ce langage ne permettait pas de donner une mnémonique à une image, alors que Usixml le permet. Cet exemple est la preuve que certains langages ne sont pas aussi vastes qu'espère l'être Usixml. Mais que se passera-t-il lorsque l'on voudra transformer le modèle concret d'Usixml vers le langage XUL, ou tout autre langage ne couvrant pas toutes les possibilités présentes et à venir d'Usixml. L'information contenue au niveau du modèle concret sera perdue.

Imaginons qu'on a une recommandation d'accessibilité, qui nous recommande d'avoir une valeur mnémonique pour les images, quand celles-ci sont transformées en version textuelle par le navigateur web. Nous aurions donc dû vérifier d'une part que les images aient une alternative textuelle, et d'autre part que la valeur mnémonique soit présente (et si possible corresponde à la première lettre de l'alternative textuelle).

Il est certain qu'on peut douter de la pertinence de donner une mnémonique à un composant image. Mais prenons l'exemple d'un menu d'une interface sur Internet qui serait composée d'image, c'est d'ailleurs, le plus souvent, le cas pour les sites asiatiques, qui affichent leur caractères via des images (cela permet de les présenter de manière plus plaisante) ou le cas pour les sites pour enfants ne sachant pas encore lire (voir par exemple le site http://www.sunisland.com.hk/enrichment.asp). Notons cependant qu'une

des recommandations d'accessibilité est de justement ne pas utiliser les images pour afficher du texte.

En quoi cela nous restreint-il ? Nous essayons de montrer qu'Usixml est un langage structuré sur base d'une hiérarchie de modèles et que valider un modèle (et seulement un) ne permet pas de s'assurer que les modèles inférieurs respecteront ces recommandations. Nous ne pouvons, dès lors, que supposer que les transformations utiliseront bien les informations contenues dans le modèle de départ.

Plus exactement, nous ne pouvons pas assurer que les informations continues dans le modèle concret qui sont nécessaires pour assurer une bonne ergonomie et accessibilité de l'interface seront bien prises en compte par les transformations pour les traduire efficacement dans l'interface utilisateur finale. Nous devons faire confiance aux règles de transformations développées (les plug-ins d'export vers le langage Java, Qtk ou HTML) pour s'assurer que les informations ajoutées dans les attributs tooltipContent des videoComponent dans le but de les convertir en des attributs *ALT* du langage HTML par exemple, soit correctement réalisées.

5.3.3 Réduction liée à l'état actuel de développement d'Usixml

Même si Usixml est destiné à s'étendre de plus en plus, à couvrir de plus en plus de modalité d'interactions et intégrer, de plus en plus, les particularités de chaque langage de programmation, Usixml a la particularité d'être tout le temps en développement. En effet, Usixml est extensible. Il est possible d'ajouter beaucoup de widgets et d'étendre à chaque fois un peu plus ses possibilités. Mais, pour l'instant, il ne couvre pas toutes les possibilités qu'offrent l'ensemble des langages de programmation. A titre d'exemple, Denis Vincent qui a développé un interpréteur d'interface utilisateur à partir de sa description (basée sur Usixml) a identifié qu'Usixml ne possède pas de toggleButton ayant la fonctionnalité d'un radioButton alors que le langage QTk possède le widget tbradiobutton. [Den05]

Dans le cadre de notre étude, certains composants sur lesquels se basent les recommandations d'ergonomie et d'accessibilité ne sont pas encore reprises dans Usixml.

Une des règles d'accessibilité, qui est pratiquement reprise par tous, concerne l'effet de clignotement qui risque de provoquer des crises d'épilepsie chez les personnes sensibles.

> *IBM – Software – Timing*
> *5.5 Do not use flashing or blinking text, objects, or other elements having a flash or blink frequency greater than 2 Hz and lower than 55 Hz.*

En regardant dans le modèle concret d'Usixml, nous ne voyons pas beaucoup d'informations concernant les animations (outre les vidéos, les gif animés cachés derrière le concept d'image et éventuellement les comportements de certains objets). Le seul composant qui donne une once d'information concernant l'emphase graphique est l'élément graphicalEmphasis qui ne contient actuellement aucun attribut. La définition qui est donnée dans la documentation d'Usixml est la suivante :

> *Enables to specify that two or more graphicalIndividualComponents must be differentiated in some way (e.g. with different color attributes)*

A part cet élément qui pourrait contenir une indication sur l'emphase particulière à ajouter lors de la conception de l'interface sur base du modèle concret, aucune information ne nous permet de vérifier que le clignotement ne peut avoir une fréquence trop élevée. Tout simplement parce que le clignotement d'un objet n'est pas spécifié dans le langage d'Usixml.

Dans le brouillon de documentation 1.4.3 d'Usixml, nous nous rendons compte que l'élément auditoryCio n'est pas encore défini. Je me permets dès lors de suggérer d'y ajouter les attributs toolTipContent et toolTipDefaultContent contenus dans graphicalCio. La présence de ces attributs pourrait nous aider à valider une des règles fondamentales de l'accessibilité qui souhaite que tout élément non textuel ait une alternative textuelle. Cette alternative textuelle devrait être renseignée dans les attributs toolTipContent au même titre que pour les éléments graphiques.

L'absence de certaines informations rend la validation totalement impossible, comme c'est le cas pour les auditoryCio. Cependant, l'absence de l'information permet parfois de valider une recommandation mais avec moins de précision. Prenons l'exemple des alternatives textuelles pour les informations non textuelles. La recommandation du WAI est la suivante :

WCAG 1.0 - 1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). This includes: *images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video.*

On peut dire qu'ils « ne font pas dans la dentelle »; tous les éléments y passent, que ce soit les images, les symboles, etc. IBM, BlindSurfer et BrailleNet ont cependant la sagesse d'affiner cette recommandation.

> *IBM – Web*
>
> 1. *Images and animations. Use the alt="text" attributes to provide text equivalents for images. Use alt="" for images that do not convey important information or convey redundant information.*
>
> *BrailleNet*
> *Les éléments graphiques destinés à la décoration sont-ils commentés par ALT="" ?*

Comment savoir maintenant que les images sont importantes ou pas : dans la version 1.4.3 d'Usixml aucune information ne nous était donnée. Nous ne pouvions dès lors que vérifier que les images, toutes les images, avaient un attribut *alt* avec un contenu non vide.

Avec la version 1.6.3, Usixml a introduit un nouvel attribut IsMandatory dans graphicalIndividualComponent. Cet attribut nous indique si le composant est obligatoire ou pas. A partir de cette information, nous supposons que si l'image n'est pas obligatoire, c'est qu'elle ne convoie pas d'informations importantes. Ainsi, avec cette information supplémentaire, nous pouvons affiner notre recommandation.

5.3.4 Limitation liée à la présence d'information dans un document Usixml

Les quatrièmes limitations observées concernent les informations présentes dans un document Usixml. Cette limitation est assez semblable à la précédente, puisqu'il est question d'absence d'information au même titre que l'absence de prise en charge de certains aspects des interfaces. Cependant, nous ne nous basons plus sur le langage Usixml, mais bien sur les informations contenues dans un document Usixml.

Une recommandation ne peut être validée que si elle s'appuie sur des attributs, informations contenues dans le document Usixml. Si le document ne contient pas ces informations, il n'est pas possible de valider la règle, ou, en tout cas, pas dans son entièreté.

Le langage Usixml est très souple et un document Usixml n'est pas obligé de contenir tous les modèles (Tâche, Domaine, Abstrait, Concret). Cependant il est possible qu'une règle ait besoin de la présence de plusieurs modèles pour être validée, nous allons d'ailleurs tenter de les indiquer lors de l'interprétation des règles.

Parmi les règles d'ergonomie, il est souvent question des concepts d'importance, de fréquence.

> *Dialogue Homme-Machine - Conception des fenêtres – Fenêtres de dialogue:*
> *Faciliter l'accès aux composants de la fenêtre les plus fréquemment utilisés*
> *Mettre en évidence les éléments les plus importants [Nog02]*
>
> *Develop system menu – Ordering [Gal96]*
> - *Order list of choices by their natural order, or*
> - *For lists with small number of options (seven or less), order by:*
>   - *Sequence of occurrence*
>   - *Frequency of occurrence*
>   - *Importance*
> - *Use alphabetic order for:*
>   - *Long lists (eight or more options).*
>   - *Short lists with no obvious pattern or frequency*

Comme nous l'avons vu au point 5.2.3.1, l'importance et la fréquence sont des attributs des tâches. Si nous avons un document Usixml qui intègre le modèle des tâches, il est alors possible de trouver un moyen pour tester et éventuellement modifier le modèle concret pour qu'il réponde aux recommandations présentées en exemple.

Si, par contre, nous n'avons pas cette information, nous pourrions cependant vérifier que les éléments du menu sont triés par ordre alphabétique par exemple. Mais cela restreint fortement la portée de la règle ergonomique.

5.4 Interprétation des règles

L'interprétation des règles pour un langage particulier n'est pas évidente. Lorsqu'on dit qu'il faut que tous les éléments d'une page web aient une valeur pour l'attribut *alt*, on ne devine pas automatiquement que c'est pour respecter une recommandation d'ordre plus général à savoir « fournir une alternatives textuelle pour tous les éléments non textuels ». L'attribut *alt* est une solution, au même titre que *longdesc*. Si nous n'avions pas ces attributs, nous aurions pu trouver une solution en ajoutant un lien hypertexte à tous les éléments non textuels vers une page qui contiendrait la description de ces éléments.

Quelles sont les recommandations que nous devons exiger d'Usixml, ou plus particulièrement d'un document Usixml, si nous voulons que ce dernier respecte les recommandations que nous avons classifiées au chapitre précédent. C'est ce que nous allons tenter de réaliser dans les deux sous-sections suivantes.

Dans un premier temps, nous nous intéresserons aux règles d'accessibilité, pour ensuite aborder légèrement les règles d'ergonomie.

5.4.1 Interprétation des règles d'accessibilité

Les règles d'accessibilité ont été classifiées en 3 catégories et différents regroupements ; nous allons essayer de voir les regroupements dans leur ensemble. Pour chaque idée, nous avons regardé quels étaient les attributs que nous devions vérifier dans Usixml pour valider cette recommandation

5.4.1.1 « Contenu » : Perception

- **Premier regroupement**

Le premier regroupement de la catégorie « Contenu – Perception » fait référence à l'idée d'alternative textuelle.

1. Fournir une alternative textuelle à tous les éléments non textuels

Les éléments VideoComponent, ImageComponent ont un attribut toolTipContent qui peut servir d'alternative textuelle. Cet attribut doit donc être renseigné pour tous les éléments graphiques.

⇒ Variante : uniquement les graphicalIndividualComponents et non les conteneurs graphiques.

L'attribut toolTipContent est un attribut commun à tous les composants graphiques. Si d'autres éléments non textuels (pas image, ni vidéo) ont besoin d'une alternative textuelle, c'est cet attribut qui sera utilisé.

Rappelons par exemple, que les composants audio, n'ont pas la possibilité d'avoir une alternative textuelle dans l'état actuel de développement d'Usixml.

⇒ Les modèles nécessaires : CUI

2. Fournir une alternative textuelle à tous les éléments importants non textuels

Première solution : Avec la version 1.6.3 d'Usixml, nous pouvons vérifier que l'attribut toolTipContent a une valeur non vide pour les éléments ayant IsMandatory mis à vrai, et une valeur vide pour tous les autres. IsMandatory serait utilisé pour indiquer que l'image est de décoration ou au contraire convoie de l'information importante.

⇒ Les modèles nécessaires : CUI

Seconde solution : Avec le modèle des tâches, il est possible de savoir si la tâche est importante ou non. S'il y a une relation de correspondance entre un élément du modèle concret et une tâche importante, nous pouvons tester avec plus de précision.

⇒ Les modèles nécessaires : T + CUI

Troisième solution : Si nous avons le modèle abstrait, il parait assez logique que les images de décoration et autres informations non pertinentes ne soient pas présentes dans ce modèle. Si l'information est reprise dans le modèle abstrait, nous considérons l'information importante, sinon non.

⇒ Les modèles nécessaires : CUI + AUI

3. Fournir des alternatives textuelles explicatives aux graphes statistiques

Les graphes statistiques et autres informations du même style ne sont pas distingués des images dans Usixml, cette recommandation est incluse dans les deux premières.

⇒ Impossible à évaluer.

4. Fournir des liens redondants pour les zones d'image

Recommandations très précises, il s'agit de voir si dans le conteneur graphique (de premier ou second niveau) de l'image, contenant la ou les zones d'image, il existe des composants textuels qui possèdent les mêmes références comme hyperlien.

Si ce n'est pas le cas, il faut transformer l'image avec ses zones d'images en un conteneur contenant l'image ainsi que la liste des hyperliens référencés par les zones d'images.

⇒ Les modèles nécessaires : CUI

5. Fournir des alternatives textuelles descriptives du contenu des éléments multimédia.

Les éléments multimédia sous Usixml ne sont catégorisés que par l'élément videoComponent. Ce dernier a les attributs subtitle et subtitleContent. Evaluer cette règle signifie vérifier que les composants vidéo ont bien des sous-titres associés.

⇒ Les modèles nécessaires : CUI

6. Synchroniser les alternatives textuelles avec les éléments multimédia.

La synchronisation est une recommandation qui s'applique plus au système qui sera chargé de présenter l'interface. Usixml est encore très statique à ce niveau.

⇒ Impossible à évaluer

7. Fournir une alternative auditive pour les éléments multimédia (live/ préenregistrée)

Il s'agirait ici de pouvoir combiner une source audio avec une source vidéo, c'est-à-dire associer un videoComponent avec un auditoryOutput. On pourrait observer si l'événement d'initiation, ou les comportements des deux éléments sont similaires. C'est-à-dire que l'événement déclenchant la lecture de la vidéo, provoque également la lecture de l'alternative audio.

⇒ Les modèles nécessaires : CUI + documentation plus précise du fonctionnement du comportement.

8. Fournir une alternative en langage des signes pour les éléments multimédia.

Nous considérons cette recommandation comme trop spécifique au rendu. En effet, le langage des signes peut-être considéré comme une représentation particulière d'un texte ou d'une piste audio. Usixml n'intègre pas encore de type d'information très spécifique.

⇒ Impossible à évaluer.

9. S'il est impossible de rendre l'interface accessible, fournir une page texte seulement.

Ce type de recommandation de dernier secours ne sera pas évalué dans Usixml. Une des solutions qui peut-être utilisée pour répondre à cette recommandation est d'utiliser un script de transformation pour produire une interface composée uniquement de texte et de joindre à l'interface un lien supplémentaire vers cette page de texte.

Cette solution ne devrait être produite qu'après l'évaluation automatique du document Usixml pour lequel un résultat d'accessibilité ne serait pas suffisant.

⇒ Pas une recommandation à évaluer, mais bien une fonctionnalité de l'outil de validation automatique.

10. Fournir une option pour afficher des indications visuelles pour les alertes audio.

Recommandation similaire à celles concernant les alternatives textuelles. La difficulté réside dans ce que l'on entend par alerte audio.

D'une certaine manière, chaque événement qui produit le lancement d'un signal audio auditoryOutput devrait également effectuer un changement dans le graphisme de l'interface, changement d'une couleur d'un objet, d'une image, etc. avec le risque de prendre en compte également les autres éléments audio qui ne sont pas des alertes.

⇒ Les modèles nécessaires : CUI + documentation plus précise du fonctionnement du comportement.

11. Fournir des informations sémantiques sur les différents objets graphiques de l'interface.

Très similaire aux alternatives textuelles, ici, il est plus question d'information sémantique, c'est-à-dire de fournir des informations qui peuvent être utilisées par les aides techniques pour aider à la perception des objets.

Usixml n'intègre pas beaucoup d'attributs au niveau du modèle concret pour donner des informations sémantiques. A l'exception du toolTipContent que nous utilisons déjà pour les alternatives textuelles, il n'existe pas d'autres attributs supplémentaires que nous pouvons utiliser pour évaluer cette recommandation.
Cela dit, l'alternative textuelle contient déjà des informations sémantiques. Mais l'on ne peut pas en donner plus.

⇒ Les modèles nécessaires : CUI + évolution prochaine d'Usixml

Dans ce premier regroupement, d'autres règles ne sont pas applicables à Usixml, pour une des raisons reprises au point 5.3. Les règles sont les suivantes :

BlindSurfer :

16 – Assurer la compatibilité avec les principaux Browsers.

⇒ Pas applicable : Dépendance vis-à-vis des interfaces web.

AccessiWeb :

1.4 Pour chacune des images de la page, les textes dans l'attribut ALT font-ils moins de 60 caractères ?

⇒ Pas applicable : Alt ne peut avoir plus de 60 caractères puisqu'il existe un autre attribut longdesc qui doit être utilisé pour les alternatives textuelles de plus de 60 caractères dans le langage HTML. C'est une spécificité du langage HTML. En Usixml nous n'avons qu'un seul attribut, l'attribut toolTipContent.

1.5 Les commentaires associés à chacune des zones réactives d'une image map sont-ils pertinents ?
1.11 Si une description détaillée de l'image est présente, son contenu est-il pertinent ?
11.6 Dans un formulaire, le commentaire du bouton SUBMIT est-il pertinent ?

⇒ Pas applicable : la pertinence est un concept qui n'est pas facilement automatisable. Plus exactement Usixml ne peut pas évaluer la pertinence d'un texte sans information complémentaire concernant le contexte, le sujet.

1.8 Pour chacune des images texte de la page, le contenu de son alternative est-il au moins équivalent au texte inscrit dans l'image ?
1.9 Il convient de remplacer un texte sous forme d'image par un texte mis en forme. Cette règle est-elle respectée?

⇒ Pas applicable : nous n'avons pas accès à l'information visuelle contenue dans l'image.

1.10 Quand une image nécessite une description détaillée, un commentaire texte lui est-il associé ?

⇒ Pas applicable : nous n'avons pas d'information concernant la nécessité d'une description détaillée pour les images. Peut-être que ce sera repris dans une version future de Usixml.

- **Deuxième regroupement**

Le deuxième regroupement de cette catégorie fait référence au principe de séparation entre le contenu et la présentation.

1. S'assurer que toutes les informations importantes qui sont transmises avec une couleur le sont également sans la couleur.

Premièrement, nous avons observé tous les endroits qui donnent une indication de la couleur. Le graphicalCio contient trois attributs fgColor, bgColor et borderColor qui indique la couleur d'avant plan, d'arrière plan et de la bordure. textComponent contient également un attribut textColor qui donne une indication sur la couleur du texte.

La recommandation peut donc être traduite de plusieurs façons différentes :

1. Si une partie de texte change de couleur (textColor, bgColor ou fgColor) par rapport au reste du texte, alors il faut que le style de cette partie de texte change aussi (isBold, isItalic, textFont, etc.)
2. Si parmi des composants adjacents, les couleurs d'avant plan et/ ou d'arrière plan changent, il faut que le style de bordure change aussi (borderWidth, borderType). L'information d'adjacence peut se faire soit au niveau du code d'Usixml, soit au niveau des relations d'adjacence d'objets.
3. Si un comportement contient une transformation qui modifie un des attributs couleur, alors une autre transformation qui modifie l'apparence (non la couleur), doit également être exécutée en même temps.

A nouveau, la notion d'importance contenue dans le modèle des tâches ou par l'attribut isMandatory nous permet d'affiner notre règle en ne l'appliquant qu'aux informations importantes.

⇒ Les modèles nécessaires : CUI (T si notion d'importance)

2. Fournir des tableaux de données facilement lisibles.

Rendre les tableaux facilement lisibles consiste à informer le mieux possible la structure des tableaux. Dans la version 1.4.3 d'Usixml, nous n'avons pas beaucoup d'informations utiles concernant les tables et les cellules. Par contre, dans la version 1.6.3 sortie dernièrement, les attributs isHeader pour les cellules et les cellules expansées nous permettent de vérifier ces règles. Il faut que chaque tableau ait une entête pour chacune de ces colonnes. C'est-à-dire que pour un élément « table » ayant y colonnes, il faut qu'il y ait y cellules (ou moins pour des cellules expansées) bien positionnées qui soient indiquées isHeader

⇒ Les modèles nécessaires : CUI

3. Utiliser des valeurs relatives au lieu de valeur absolue dans les attributs de mise en page.

Dans Usixml, les valeurs sont souvent relatives, comme par exemple relativeWidth et relativeHeight de l'élément box. Les valeurs qui ne sont pas relatives sont les suivantes :

1. windowLeftMargin, windowRightMargin et textMargin mais qui n'influence pas fortement le rendu de l'interface.
2. width et height des conteneurs graphiques qui ne précisent pas dans la documentation si les valeurs sont relatives ou absolues.

3. textSize des textComponent qui exprime la taille de la police en points. C'est une valeur absolue.
4. imageHeight, imageWidth, imageBorder qui exprime la taille d'une image. Ces valeurs sont absolues mais sont justifiables puisque l'image à une dimension fixe.

Le langage Usixml n'est pas aussi souple que l'HTML, c'est-à-dire que soit l'information est donnée en valeur relative, soit en valeur absolue. Nous n'avons pas le choix. Cette recommandation n'est donc pas évaluable.

⇒ Pas applicable à Usixml

4. Organiser le document de sorte qu'il puisse être lu sans feuille de style.

C'est une recommandation qui est reprise par beaucoup de sources et illustre bien le souci de la séparation entre le contenu et la présentation. Malheureusement ce type de recommandation est trop typique au langage HTML.

⇒ Pas applicable à Usixml

Outre ces règles, nous en avons d'autres qui ne sont pas applicables à Usixml.
Soit parce qu'elles sont trop liées au langage HTML.

WCAG

3.5 Mark up lists and list items properly.
3.6 Mark up quotations. Do not use quotation markup for formatting effects such as indentation.
5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting.
5.5 Provide summaries for tables.

AccessiWeb

5.1 L'attribut SUMMARY est-il présent et pertinent ?
5.2 Dans un tableau de données, la balise CAPTION est-elle utilisée pour donner un titre au tableau ?
13.8La présentation de la page est-elle réalisée sans détourner certaines balises de leur fonction d'origine ?

Recommandation trop liée au langage HTML, d'autant plus que nous n'avons pas le moyen de d'indiquer un résumé ou une légende pour un tableau dans Usixml.

Soit parce qu'elles expriment l'indépendance du contenu vis-à-vis de la présentation par des conseils techniques dépendant du langage de programmation cette fois.

508

Textual information shall be provided through operating system functions for displaying text. The minimum information that shall be made available is text content, text input caret location, and text attributes. (Software – f)

IBM

Implement the Java Accessibility API by:
- using the Java Foundation Classes (JFC) / Swing components and/or
- following the guidelines for " Building Custom Components" when extending

the Java Foundation Classes and when implementing the Java Accessibility API on custom components. (Java 2.1)

Provide text through standard system function calls or through an API (application programming interface) which supports interaction with assistive technology. (Software 4.1)

Support system settings for size, font and color for all user interface controls. (Java 4.4 / Software 4.5]

- **Troisième regroupement**

Le troisième et dernier regroupement de cette catégorie parle plus explicitement des couleurs et de leur contraste.

1. ***S'assurer que les combinaisons de couleurs d'avant plan et d'arrière plan fournissent un contraste suffisant pour être vu par des personnes ayant des déficiences dans la perception des couleurs.***

A nouveau, les seuls outils qui nous informent sur les couleurs sont fgColor, bgColor, borderColor et textColor.

En ce qui concerne les bonnes combinaisons de couleur, nous pouvons nous aider de l'étude [Vand04] qui a évalué le même type de règle pour les sites web. En ce basant sur les bonnes combinaisons de couleurs proposées par Murch. L'application dans Usixml de cette règle s'applique à différents cas.

1. La combinaison fgColor et bgColor dans un même composant.
2. Les combinaisons fgColor et borderColor et les combinaisons bgColor et borderColor d'un même composant, si l'attribut borderWidth est non nul.
3. Les combinaisons fgColor et textColor et les combinaisons bgColor et textColor pour un même composant texte.

Viennent ensuite les hiérarchies, c'est-à-dire l'attribut d'un fgColor d'un conteneur avec l'attribut fgColor d'un autre conteneur contenu dans le précédent. Ou l'attribut fgColor d'un conteneur avec le textColor d'un composant texte contenu dans le conteneur. Et ainsi de suite pour toutes les possibilités.

⇒ Les modèles nécessaires : CUI

2. ***Lorsque l'interface permet à l'utilisateur de choisir la configuration des couleurs et contrastes, une variété de couleurs de sélection permettant de produire un haut niveau de contraste doit être fournie.***

Cette recommandation n'est pas évaluable dans Usixml. Bien qu'il y ait un élément colorPicker, ce dernier ne contient aucune information sur la quantité de couleur, ni sur les contrastes des couleurs qui sont possibles. C'est un objet abstrait, il prendra différents formats sur des plates-formes différentes ou des langages différents.

⇒ Pas applicable à Usixml.

5.4.1.2 « Contenu » : Compréhension

- **Premier regroupement**

Le premier regroupement que nous avons dégagé concerne le langage utilisé dans l'interface et insiste sur le besoin de clarifier l'usage du langage naturel.

1. ***Clarifier clairement les changements du langage naturel dans le document.***

   Usixml n'a pas au niveau concret d'indication sur le type de langage naturel utilisé. C'est une limitation liée à l'état de développement actuel.

   ⇒ Pas applicable à Usixml.

2. ***Expliquer la signification de chaque abréviation, acronyme du document lorsqu'il apparaît pour la première fois.***

   Encore une fois, Usixml n'a pas un niveau concret d'indication sur les abréviations ou les acronymes. C'est également une limitation liée à l'état de développement actuel.

   Cependant, il est possible de faire un balayage des textes pour tenter de retrouver des abréviations (suite de caractères n'appartenant pas au dictionnaire suivi d'un point) ou des acronymes (suite de caractères en majuscule n'appartenant pas au dictionnaire) et de les indiquer à l'utilisateur superviseur de l'évaluation. Mais pour cela, il faut utiliser des moteurs de recherche linguistiques.

   ⇒ Pas applicable à Usixml.

- **Deuxième regroupement**

Le deuxième regroupement est très spécifique à des aides supplémentaires pour comprendre le sujet de l'interface.

1. ***Ajouter des textes, des graphes, des présentations audio où ils facilitent la compréhension de la page.***

   Malheureusement ce type de recommandation n'est pas applicable à Usixml dans le cadre d'un outil d'évaluation automatique. La compréhension d'une page n'est pas quelque chose de facile à évaluer et il n'est pas évident de savoir quelle information ajouter. Cela dit, l'ajout d'icônes à chaque menu, de graphique pour des tableaux de données, sont des variantes qui peuvent être éventuellement évaluées. Mais nous remarquerons dans les règles d'ergonomie que, concernant les icônes, il n'est pas bon d'en avoir de trop.

   ⇒ Pas applicable à Usixml.

- **Troisième regroupement**

Le troisième regroupement parle de consistance de l'interface, de la cohérence d'une interface.

1. ***L'information des objets doit être constante dans l'interface.***

   - Les composants de textes qui ont le même hyperlien, doivent avoir le même contenu textuel.
   - Les couleurs des composants de textes ayant un hyperlien, ainsi que les couleurs des liens actifs et visités doivent être les mêmes dans tout le document.
   - La taille des composants images (ayant la même image source) doit être identique.

- Les images contenant des hyperliens identiques, doivent avoir le même texte alternatif (indiquant l'information du lien).
- Chaque type de widget (radiobutton, togglebutton, checkbox, …) doit avoir le même look, même couleur, même bordure, etc.
- Les widgets identiques doivent avoir les mêmes raccourcis clavier, et les mêmes mnémoniques.

⇒ Les modèles nécessaires : CUI

2. Créer une interface qui reste prévisible, logique.

Cela concerne fortement les comportements des objets qui doivent fonctionner de manière constante, il en va de même pour la navigation. Les éléments du modèle concret qui ont une facette de navigation dans le modèle abstrait doivent avoir un comportement identique. Tous les objets qui ont des comportements dont les événements sont de type depress et release, dragOver et dragDrop, hasFocus et lostFocus doivent avoir des actions réciproques, c'est-à-dire que si hasFocus rend le bouton de couleur bleue, l'action lostFocus remet le bouton dans sa couleur initiale. Les items contenues dans les comboBox, les spins, les trees doivent avoir un ordre logique (alphabétique, alphanumérique, numérique, autre indication donnée par l'utilisateur superviseur). Si un objet est particulièrement important et utilisé fréquemment, il faut vérifier que sa position dans l'interface est constante.

L'utilisateur doit être prévenu de l'ouverture d'une nouvelle fenêtre, et l'ouverture de fenêtre pop-up doit être proscrite. En Usixml, il faut qu'un événement utilisateur soit effectué pour ouvrir une fenêtre.

⇒ Les modèles nécessaires : CUI + AUI + éventuellement intervention de l'utilisateur superviseur.

3. Créer un style constant, cohérent dans l'interface utilisateur.

C'est-à-dire que la structure des conteneurs imbriqués, des tableaux, des widgets doit être la plus constante possible. Une analyse de l'imbrication des balises du langage Usixml, permet de comparer la structure de différentes fenêtres, conteneurs, etc. Si par exemple, un conteneur graphique de type fenêtre contient toujours deux box, alors si une fenêtre contient 3 box, un message sera envoyé à l'utilisateur pour lui signaler une non consistance. La position du menu dans une fenêtre doit être constante. C'est-à-dire que l'imbrication des balises entre la balise ouvrante de la fenêtre et la balise du menu doit être la même.

⇒ Les modèles nécessaires : CUI.

4. Créer des objets faciles à comprendre

C'est-à-dire des données, des informations concises et précises. Par exemple, de fournir des hyperliens concis (moins de 80 caractères selon AccessiWeb). Le contenu du textComponent qui a son attribut hyperlien indiqué ne doit pas avoir plus de 80 caractères. Une autre règle d'AccessiWeb (6.2) demande que les liens soit explicites. A ce niveau il est parfois conseillé d'indiquer comme texte du lien, le titre de l'interface qui est référencé par l'hyperlien. Dans Usixml, si le lien fait référence à une information contenue dans le document, telle qu'une autre fenêtre, alors c'est évaluable. Mais si le lien fait référence à une information en dehors du document (à

Internet par exemple), alors il y a d'autres moyens d'évaluation comme par exemple, l'utilisation d'un navigateur pour récupérer le titre de la page.

⇒ Les modèles nécessaires : CUI

5.4.1.3 « Navigation »

- **Premier regroupement**

Le premier regroupement que nous avons réalisé sous le critère de navigation est l'indépendance par rapport à l'outil de navigation (clavier, souris, …).

1. Rendre les objets activables par le clavier.

En Usixml, les attributs qui seront utilisés pour vérifier l'accessibilité par le clavier sont keyboardShortcut et defaultKeyboardShortcut des menuItem et les attributs defaultMnemonic et mnemonic des graphicalIndividualComponent, ainsi que l'attribut index des éléments tabbedItem. Les règles interprétées pour Usixml peuvent s'exprimer comme suit :

1. Tout élément menuItem doit avoir des valeurs non vides pour ses attributs keyboardShortcut et defaultKeyboardShortcut.
2. Tout élément graphicalIndividualComponent doit avoir des valeurs non vides pour ses attributs defaultMnemonic et mnemonic.
3. Tout élément tabbedItem doit avoir une valeur unique pour l'attribut « index » à l'intérieur d'un même tabbedDialogBox

⇒ Les modèles nécessaires : CUI

2. Variation : Rendre les objets importants activables par le clavier.

Si nous avons l'information d'importance, de fréquence, permettant d'identifier les objets importants des autres. On peut affiner la règle précédente pour ne l'appliquer qu'aux objets qui sont importants ou utilisés fréquemment. Usixml contient des informations sur l'importance et la fréquence dans le modèle des tâches. Mais également dans le modèle CUI par l'attribut isMandatory apparu dans la version 1.6.3.

⇒ Les modèles nécessaires : CUI + T

3. Les raccourcis claviers et mnémoniques ne doivent pas interférer avec le système de la plate-forme.

Plus exactement, cela ne doit pas interférer avec les mnémoniques et raccourcis claviers par défaut du système. Les informations concernant la plate-forme ne sont pas présentes dans le CUI, mais sont parfois renseignées dans le modèle de contexte. Si on a une liste (ajoutée par l'utilisateur superviseur) des différents raccourcis claviers et mnémoniques du système de la plate-forme de destination. L'interprétation de cette règle pour Usixml serait la suivante :

1. Tout élément ayant un raccourci clavier et/ou un mnémonique renseigné, ce dernier doit être différent de ceux contenus dans la liste.

⇒ Les modèles nécessaires : CUI + Contexte/Superviseur.

- **Deuxième regroupement**

Le second regroupement concerne les notions de temps, temps de réaction, temps de délai, temps de réponse.

1. ***Permettre à l'utilisateur de contrôler le temps de réponse ou laisser les instructions persister.***

   Le modèle des tâches donne une indication concernant la succession des tâches et indique si une tâche s'exécute de manière automatique après un certain laps de temps. On peut s'assurer qu'aucune tâche ne soit interrompue automatiquement, c'est-à-dire sans l'intervention de l'utilisateur, grâce aux éléments disabling et suspendResume et de l'attribut userAction des éléments task.

   ⇒ Les modèles nécessaires : Tâches.

   L'autre possibilité d'indication de « temps » est celle des comportements contenus dans le CUI. Normalement aucun comportement n'est automatique. Tous doivent être actionnés par un utilisateur. Sans action de l'utilisateur, les instructions persistent. Le modèle CUI vérifie intrinsèquement cette recommandation.

   ⇒ Vérifié intrinsèquement.

- **Troisième regroupement**

Le troisième regroupement concerne les phénomènes de clignotements et du risque de crise d'épilepsie qu'ils peuvent provoquer.

1. ***Eviter les animations.***

   Dans le modèle concret d'Usixml, les animations sont présentes dans les videoComponent et les imageComponent. Dans les imagesComponent l'animation peut être représentée par une image animée (un gif animé par exemple), mais aucune indication sur la nature de l'image n'est donnée. Donc nous ne pouvons pas tester les imageComponent. Dans les videoComponent, l'attribut alternateImage est spécialement prévu pour donner une image si le système n'est pas en mesure d'exécuter la vidéo.
   Une première interprétation de la règle est que tous les videoComponent doivent avoir une valeur non nulle pour leur attribut alternateImage.

   ⇒ Les modèles nécessaires : CUI

2. ***Eviter le clignotement***

   Le clignotement est une technique de mise en évidence qui n'est pas prise en charge par le langage d'Usixml. Finalement les seuls éléments d'Usixml qui peuvent encore perturber l'utilisateur dans l'interface finale, sont les graphicalEmphasis mais qui n'ont pas encore d'attributs dans l'état actuel de développement d'Usixml, et ne sont que des indications de mise en évidence dans le langage naturel.

   ⇒ Pas applicable à Usixml.

3. ***Eviter le mouvement dans l'interface***

   Le mouvement n'est pas renseigné dans le langage Usixml et il semble difficile d'en créer un en utilisant uniquement les comportements des objets.

   ⇒ Pas applicable à Usixml.

4. Donner la possibilité à l'utilisateur d'arrêter, de contrôler les animations

Les animations ne sont donc représentées que par les videoComponent. Si l'attribut autoplay est mis à vrai ou si l'attribut loop est mis à vrai, il faut qu'il y ait un élément pour lequel une action est de stopper le composant videoComponent, ou de mettre l'attribut loop à faux.

⇒ Les modèles nécessaires : CUI.

- **Quatrième regroupement**

Le dernier regroupement concerne les mécanismes de navigation offerts.

1. Fournir un titre différent à chaque fenêtre, à chaque composant.

Dans le modèle concret d'Usixml, tous les objets héritent de l'attribut name de cio. Tous les objets doivent avoir une valeur non vide et différente pour cet attribut.

⇒ Les modèles nécessaires : CUI.

2. Indiquer à tout instant quel est l'objet qui possède le focus.

Dans Usixml, au niveau des comportements, le type d'événement peut être du type hasFocus. Il faut donc que, pour chaque objet, il y ait un comportement associé avec le type d'événement hasFocus et lostFocus.

⇒ Les modèles nécessaires : CUI

3. Donner la possibilité à l'utilisateur d'accéder directement à l'information sans devoir passer par différentes pages ou fenêtres d'introduction.

Usixml ne donne pas d'information permettant de faire la distinction entre des fenêtres d'introduction et les autres fenêtres. Que ce soit au niveau CUI, AUI.

Au niveau tâches, il est possible de voir si l'utilisateur peut accéder directement à une autre tâche. C'est-à-dire voir s'il existe des raccourcis.

⇒ Les modèles nécessaires : Tâches.

4. Fournir de la redondance dans la navigation, plusieurs liens identiques, plusieurs boutons de navigation.

C'est-à-dire qu'une fenêtre doit inclure plusieurs fois les mêmes textComponent ayant des liens identiques.

⇒ Les modèles nécessaires : CUI

5. Fournir une barre de navigation ou un menu dans l'interface.

Toutes les fenêtres du modèle CUI doivent avoir un élément « menu » attaché.

⇒ Les modèles nécessaires : CUI

6. Dans les formulaires, est-ce que les champs de même nature sont entourés d'une bordure munie d'un titre ?

Il est recommandé que les checkboxs, les radiobuttons, les champs éditables, etc. soient contenus dans une box et que cette box aient une bordure visible (borderWidth > 0) et un titre (Valeur non vide pour l'attribut borderTitle).

⇒ Les modèles nécessaires : CUI

7. Séparer par des caractères, les liens hypertextes adjacents.

Afin de rendre la navigation plus claire, il est bon que des liens ne soient pas « collés » entre eux. Deux textComponent ayant leurs attributs hyperlinkTarget non vides doivent contenir un autre textComponent entre eux qui n'ait pas de valeur non vide pour l'attribut hyperlinkTarget.

⇒ Les modèles nécessaires : CUI

8. Est-ce que les labels sont bien mis en correspondance avec leur champ d'édition, et positionnés de façon claire et non ambiguë ?

Usixml permet d'indiquer des relations d'adjacence (graphicalAdjacency), ainsi que des alignements (alignement) pour mettre en correspondance les labels et leur champ d'édition. Pour savoir si le label et le champ d'édition sont bien mis en correspondance, nous pouvons faire une analyse sur le contenu du texte du label (le contenu du textComponent) et le nom de l'attribut dans le domaine qui est en relation avec le champ d'édition. Si les contenus correspondent, nous pouvons être presque certains qu'ils sont bien mis en correspondance.

⇒ Les modèles nécessaires : CUI + Domaine

9. Fournir un moteur de recherche interne à l'application.

Un moteur de recherche c'est une fonctionnalité de l'application, pas une propriété de l'interface graphique.

⇒ Pas applicable à Usixml

10. Pour les champs de formulaires, donner une valeur par défaut des champs ou un texte tenant place.

C'est-à-dire que pour tout textComponent dont l'attribut isEditable est mis à vrai, doit avoir une valeur non vide pour son attribut content.

⇒ Les modèles nécessaires : CUI

Si l'attribut content est vide, nous pouvons déjà proposer une modification qui remplira l'attribut « content » par la valeur contenue dans attributeInstance du modèle du domaine pour l'objet en correspondance.

⇒ Les modèles nécessaires : CUI + Domaine

11. Si l'interface nécessite qu'une technologie particulière soit installée sur la plate-forme pour être exécuté correctement, il faut donner la possibilité à l'utilisateur de l'installer.

Usixml est indépendant de la technologie qui sera employée pour produire l'interface finale. Si cette dernière a besoin d'une technologie particulière tel que flash, c'est à cette dernière qu'il est recommandé d'ajouter la possibilité à l'utilisateur d'installer cette technologie.

⇒ Pas applicable à Usixml

Nous voilà à la fin de notre interprétation des règles d'accessibilités, comme nous le prévoyions, il n'est pas possible d'interpréter toutes les règles à Usixml. Dans la sous-

section suivante, nous allons tenter d'interpréter les règles d'ergonomie pour le langage Usixml.

5.4.2 Interprétation de quelques règles d'ergonomie choisies

Vu la quantité importante de règles, 944 chez Smith et Mosier, plus de 300 chez Galitz, il est certain qu'il n'est pas possible d'interpréter toutes les règles. Nous allons cependant en présenter quelques-unes pour chacun des critères présentés au chapitre précédent.

5.4.2.1 Guidage

5.4.2.1.1 Incitation

1. ***Indiquer le but de l'interface, à quoi sert l'interface***

En supposant que les informations contenues dans le modèle des tâches sont correctes, le nom de la tâche qui indique le sujet de la tâche devrait être identique à l'entête de la fenêtre finale ou d'une boite finale. C'est-à-dire que tout conteneur graphique sémantiquement relié à un conteneur abstrait, lui-même relié à une tâche, doit contenir comme titre (attribut content) le même texte que l'attribut name de la tâche dans le modèle de tâches.

⇒ Les modèles nécessaires : CUI + AUI + Tâches.

2. ***Positionner les éléments les plus importants le plus haut possible.***

Pour avoir une notion d'importance, nous avons besoin du modèle des tâches. Dès que nous avons nos tâches importantes qui sont matérialisées sous forme de conteneurs graphiques dans le modèle concret. Et que dans un conteneur fenêtres, nous avons un ou plusieurs conteneurs graphiques importants, leur ordre doit respecter la graduation de l'importance (niveau de 1 à 5) identifiée dans le modèle des tâches.

⇒ Les modèles nécessaires : CUI + AUI + Tâches.

5.4.2.1.2 Groupement/distinction entre items

1. ***Utiliser des couleurs très contrastées pour exprimer une différence, choisir des couleurs peu contrastées pour exprimer une similarité.***

Qu'est ce qui est différent, qu'est ce qui est similaire ? Nous n'avons pas d'indication dans le langage Usixml sur les éléments qui sont similaires ou différents. Nous pouvons nous baser sur une observation des attributs pour des éléments identiques. C'est-à-dire que si deux check-box ont des attributs, des positions similaires, alors leurs couleurs ne doivent pas être contrastées. Par réciprocité, si les attributs, positions sont très différentes, alors leurs couleurs doivent être contrastées. Sans des informations claires sur les similitudes et différences, qui seraient données par l'utilisateur superviseur, il n'est pas possible de s'assurer que le résultat de l'évaluation soit correct.

⇒ Les modèles nécessaires : CUI (+ Utilisateur superviseur).

2. ***Regrouper les éléments de navigation, ensemble.***

Les éléments de navigation sont les objets du modèle concret qui sont en correspondance avec un objet du modèle abstrait pour lequel il existe une facette de type navigation. Les regrouper, signifie que les objets ayant une facette de type navigation soient tous, et uniquement ceux là, inclus dans un conteneur abstrait.
⇒ Les modèles nécessaires : AUI

5.4.2.1.3 Retour utilisateur

1. ***Signaler tous événements par un changement graphique, ou sonore.***

Toute action de l'utilisateur doit se traduire par une modification de l'interface au niveau graphique ou sonore. Dans Usixml, il faut que chaque événement pris en compte par un objet ayant une modification graphique et/ou sonore de l'interface. Une modification c'est-à-dire une action de transition ou de transformation. Dans le cas où l'action est de type appel de méthode, modifier le curseur en un indicateur d'attente correspond également à joindre une action de transformation au comportement de l'objet.
⇒ Les modèles nécessaires : CUI

5.4.2.1.4 Lisibilité

1. ***Les textes doivent être affichés en minuscules, la première lettre étant en majuscule.***

L'interprétation à Usixml, consiste à dire que tous les composants textes visibles, doivent être écrits en minuscule, la première lettre en majuscule. Le contenu des textComponents, des boutons, des radionbuttons, des checkbox, des spins, des comboBox, des items, etc… doivent correspondre à cette règle
⇒ Les modèles nécessaires : CUI

5.4.2.2 Charge de travail

5.4.2.2.1 Brièveté

1. ***Les libellés doivent être concis.***

La concision n'est pas quelque chose de facile à définir. Nous pouvons cependant vérifier que les libellés comptent moins de X caractères. X étant une variable que le concepteur peut indiquer.
Le contenu des textComponents, des boutons, des radiobuttons, des checkbox, des spins, des comboBox, des items, etc… doivent être de moins de X caractères.
⇒ Les modèles nécessaires : CUI

2. ***Les listes d'items doivent être courtes (7±2 items)***

Il faut que les widgets spin, comboBox et tree qui sont composés d'items doivent en contenir 7±2 items, c'est-à-dire pas moins de 5, pas plus de 9.
Les valeurs 7±2 sont des valeurs qui peuvent être modifiées par l'utilisateur superviseur.
⇒ Les modèles nécessaires : CUI

5.4.2.2.2 Densité informationnelle

1. ***Les images de décoration doivent être peu nombreuses.***

Comme nous l'avons montré dans l'interprétation des règles d'accessibilité, les images de décoration sont celles dont l'attribut isMandatory est mis à faux. Leur nombre et leur taille doit être le plus faible possible en comparaison des autres images. Par un algorithme qui calculerait la surface proposée par les différents ensembles, on serait en mesure de savoir si les images de décoration sont surabondantes ou non.
La somme des tailles (hauteur * largeur) de toutes les imageComponent dont l'attribut isMandatory est à faux (ou vide), doit être plus faible que la somme des tailles de tous les imageComponent dont l'attribut isMandatory est à vrai.
⇒ Les modèles nécessaires : CUI

2. ***N'utiliser pas plus de X familles de polices de caractères, compatibles en termes d'épaisseur de traits, etc.***

Les attributs textFont des éléments inclus dans un conteneur graphique de type window ne doivent pas être trop différents, pas plus de X familles différentes.
X est une variable qui doit être donnée par l'utilisateur superviseur.
⇒ Les modèles nécessaires : CUI

3. ***N'utiliser pas plus de Y tailles différentes pour les textes.***

Les attributs textSize des éléments inclus dans un conteneur graphique de type window ne doivent pas être trop différents, pas plus de Y tailles différentes. Y est une variable qui doit être donnée par l'utilisateur superviseur. Si possible 12 points pour les menus, et 10 points pour les fenêtres.
⇒ Les modèles nécessaires : CUI

5.4.2.3 Contrôle explicite

5.4.2.3.1 Action explicite

1. ***Valider explicitement les commandes destructrices***

Les commandes destructives seront sans doute spécifiées dans l'attribut action de l'élément control du modèle abstrait. Cette information n'est pas encore présente dans la version 1.4.3, ni dans la version 1.6.3.
⇒ Pas applicable à Usixml

5.4.2.3.2 Contrôle utilisateur

1. ***Autoriser les retours en arrière.***

La présence d'un élément de menu nommé « Undo » ou « Annuler/Défaire » selon la langue permet de vérifier qu'il est possible de revenir en arrière. Normalement le comportement de ce menu doit faire appel à une méthode. Le menu de chaque fenêtre doit avoir un élément de menu nommé ainsi.
⇒ Les modèles nécessaires : CUI + éventuellement le modèle du Domaine.

5.4.2.4 Adaptabilité

5.4.2.4.1 Flexibilité

1. Voir les règles d'accessibilité

5.4.2.4.2 Le prise en compte de l'expérience de l'utilisateur

1. Permettre un accès rapide et direct aux commandes fréquentes par des raccourcis clavier.

Tous les objets déclenchant des tâches identifiées comme fréquentes dans le modèle des tâches doivent avoir des raccourcis clavier. Notons que seuls les menus ont des informations concernant les raccourcis clavier. Le déclenchement des tâches est indiqué dans le comportement des objets du modèle concret qui doivent être mis en correspondance avec le modèle de tâche.

⇒ Les modèles nécessaires : CUI + Tâches (+ AUI).

Si le lien entre le menu et la tâche fréquente n'est pas possible, on peut demander à ce que tous les menus aient des raccourcis clavier.

⇒ Les modèles nécessaires : CUI.

5.4.2.5 Gestion des erreurs

5.4.2.5.1 Protection contre les erreurs

1. Fournir la liste des valeurs permises, minimiser les saisies au clavier.

Si nous avons le modèle du domaine et que des valeurs possibles sont énumérés pour un attribut X de l'objet Y, et que l'attribut X est en correspondance avec un type d'objet input du modèle abstrait, alors l'objet Z du modèle concret qui concrétise l'objet input du modèle abstrait doit être du type comboBox, contenant pour items les valeurs énumérées dans le domaine du modèle.

⇒ Les modèles nécessaires : CUI + AUI + Domaine.

2. Séparer clairement les commandes destructrices des autres en les plaçant en bas des menus.

Comme nous l'avons vu au point 5.4.2.3.1, nous n'avons pas dans l'état actuel de développement d'Usixml, d'indication sur les commandes destructrices.

⇒ Pas applicable à Usixml.

Si nous avions ces informations, nous pourrions observer leur position par rapport aux autres dans l'ordre des menus.

⇒ Les modèles nécessaires : CUI + AUI.

5.4.2.5.2 Qualité des messages d'erreur

1. Utiliser le langage de l'utilisateur, de la tâche.

Comme nous l'avons vu dans la partie des règles d'accessibilité, Usixml ne donne pas d'indication sur le langage naturel de l'utilisateur. Concernant le langage de la tâche, non plus. Si Usixml nous donne ces informations, il est alors possible d'utiliser des dictionnaires particuliers pour le sujet de la tâche qui contiendrait les mots adaptés à la tâche.

⇒ Pas applicable à Usixml.

5.4.2.5.3 Correction des erreurs

1. ***Fournir la possibilité de modifier facilement les données saisies, une fois l'erreur signalée.***

Pour s'assurer que les données saisies sont modifiables, on doit vérifier qu'aucune action de modifie l'attribut isEditable d'un champ éditable.

⇒ Les modèles nécessaires : CUI

5.4.2.6 Homogénéité / Cohérence

1. ***Adopter un vocabulaire homogène.***

Il est préférable d'utiliser toujours les mêmes mots, quitte à les répéter plusieurs fois. L'interprétation pour Usixml serait que tous les contenus synonymes de différents objets doivent être remplacés par un et un seul des synonymes.

⇒ Les modèles nécessaires : CUI + dictionnaire des synonymes de la langue voulue.

2. ***Placer les boutons de contrôle à une position cohérente dans l'application.***

La cohérence n'est pas chose facile à évaluer. Il faut donc que les boutons de contrôle soit toujours au même endroit dans les différentes fenêtres. Classiquement on dit qu'il faut que les boutons de contrôles soit en bas de la fenêtre. L'interprétation pour Usixml serait que tous les groupements de boutons de contrôles doivent se situer à une même position à travers les fenêtres. C'est-à-dire, tout le temps à droite, à gauche, en haut ou en bas.

⇒ Les modèles nécessaires : CUI

5.4.2.7 Signification des codes et dénominations

1. ***La mnémonique doit être le premier caractère de la description de l'élément de menu. S'il y existe une duplication, utiliser une autre lettre pour les mnémoniques, si possible la première consonne suivante.***

Vérifier que la valeur des attributs mnemonic des éléments menuItem corresponde à la première lettre du contenu indiqué pour ces éléments.

⇒ Les modèles nécessaires : CUI

2. ***Les icônes doivent être tout différentes les unes des autres.***

Vérifier que les valeurs des attributs icon de tous les éléments soient toutes différentes à l'intérieur d'une même fenêtre. Tous les éléments sont pris en compte puisqu'ils héritent tous de l'attribut icon par le cio.

⇒ Les modèles nécessaires : CUI

5.4.2.8 Compatibilité

1. Respecter le sens que l'utilisateur donne aux couleurs.

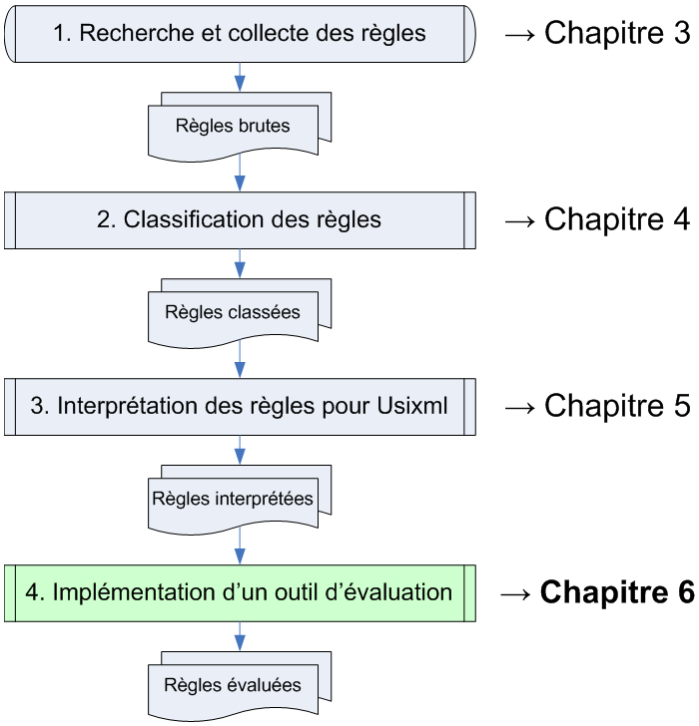
Nous pouvons, à partir d'heuristiques et d'informations supplémentaires, savoir le sens que l'utilisateur donne à chaque couleur. Mais nous n'avons pas beaucoup d'informations dans Usixml sur le sens donné à chaque objet. L'importance de certains objets est donnée par isMandatory, mais pas beaucoup plus. Nous ne savons pas si un textComponent correspondant à un champ d'un formulaire est optionnel ou obligatoire.

⇒ Pas applicables à Usixml.

Chapitre 6 Implémentation / Evaluation des règles

Nous avons interprété les règles pour le langage particulier, c'est-à-dire Usixml dans notre étude, dans le chapitre précédent. Maintenant que nous avons un ensemble de règles interprétées, nous pouvons effectuer la dernière étape de notre méthodologie, c'est-à-dire l'évaluation de ces règles, ou plus exactement étudier quelles techniques sont possibles pour évaluer automatiquement ces règles.

Dans une première section, nous présenterons différentes techniques d'évaluation possible. Dans la section 6.2, nous présenterons la transformation des règles interprétées en des règles évaluables automatiquement via une technique d'évaluation. La section 6.3 présentera différentes suggestions d'intégration à des logiciels existants.



6.1 Techniques d'évaluation

Il existe différentes techniques pour évaluer des règles d'ergonomie et d'accessibilité.
Dans le cadre d'Usixml, les techniques de validation sont principalement des techniques issues des études sur les documents XML, puisque le langage d'Usixml est compatible, mais aussi des techniques utilisés pour évaluer les documents HTML. L'HTML est aussi un langage de marquage, composé de balises et d'attributs.

L'étude de Vicente Luque Centeno [Cen05] intitulé « WCAG Formalization with W3C Techniques » reprend différentes techniques du W3C applicables pour l'évaluation des règles du WCAG, et donc des documents (X)HTML. Nous avons décidé de reprendre les différentes techniques qu'ils proposent et d'expliquer en quoi ces techniques peuvent aider à l'évaluation des règles pour Usixml.

- **Evaluation grâce à la grammaire du langage**

Il s'agit de l'évaluation des règles qui sont intrinsèquement vérifiées par la grammaire du langage. Par exemple, il est déconseillé de faire clignoter un texte puisqu'il risque, pour certains groupes de personnes, de provoquer des crises d'épilepsie. Mais puisque le langage Usixml ne permet pas d'indiquer le clignotement d'un texte, la règle est intrinsèquement vérifiée par la grammaire, par la structure du langage Usixml.

  - Avantage : Les règles sont intrinsèquement vérifiées
  - Inconvénient : Il n'y a pas réellement d'évaluation, cela limite fort l'ensemble des règles évaluables.

- **Evaluation grâce à une grammaire déclarée**

Ici, il ne s'agit plus de la grammaire de base du langage, mais d'une grammaire explicitée par l'usage d'un schéma XML. Par exemple, une des interprétations que nous avons effectuées pour le langage Usixml dit que tous les videoComponent doivent avoir leur attributs subtitle et subtitleContent. L'usage d'un schéma XML plus stricte que celui de base permet d'exiger la présence d'un élément ou d'un attribut. Cette technique est très bon marché, ainsi que très simple. Pour l'évaluation, il suffit de valider le document Usixml grâce à un outil de validation grammaticale comme W3C Markup Validation Service [W3C05] ou Validateur HTML/XHTML [Mor05].

  - Avantage : traduction des règles facile et pas cher du tout.
  - Inconvénient : C'est limité à des évaluations basées sur la présence ou absence d'attributs et d'éléments.

- **Evaluation grâce à des règles exprimées en XPath 1.0.**

L'évaluation par la grammaire déclarée permet de tester l'absence ou la présence d'élément ou d'attribut. C'est tout blanc ou tout noir. L'évaluation par des règles exprimées en XPath [XPa99] permet de signaler que certains élément ne sont pas recommandés malgré qu'ils ne soient pas interdits pour autant, ou de signaler des combinaisons d'attributs interdites. C'est le cas, par exemple, pour les attributs isMandatory et toolTipContent d'un composant image par exemple. Soit isMandatory est à vrai, et le toolTipContent a une valeur autre que la chaîne vide, Soit isMandatory est à faux et le toolTipContent a une valeur vide.

- Avantage : Les règles sont bon marché et relativement facile à exprimer.
- Inconvénient : Il nécessite un support, un moteur implémenté pour pouvoir l'utiliser.

- **Evaluation grâce à des règles exprimées en XQuery 1.0.**

Pour toutes les règles qui font intervenir plusieurs balises, ou des relations entres deux ou plusieurs éléments d'un document, comme c'est, précisément, le cas pour l'ensemble des règles interprétées qui font intervenir plusieurs modèles d'Usixml, XPath n'est pas capable d'exprimer les règles. On a besoin de faire appel à XQuery [XQu05]. Ce dernier permet d'exprimer des requêtes semblables à SQL comme l'illustre la Figure 19.

- Avantage : Très puissante, cette technique permet d'exprimer des règles nettement plus complexes.
- Inconvénient : N'est pas aussi puissante que SQL pour la recherche d'information, ne contient pas de « Group by » par exemple.

```
for $d in document("depts.xml")//deptno
let $e := document("emps.xml")//employee[deptno = $d]
where count($e) >= 10
order by avg($e/salary) descending
return
  <big-dept>
     { $d,
        <headcount>{count($e)}</headcount>,
        <avgsal>{avg($e/salary)}</avgsal>
     }
  </big-dept>
```

Figure 19 - Un exemple complet d'une requête en XQuery

- **Evaluation grâce à des règles basées sur XPointer.**

Dans l'étude de Vicente Luque Centeno, il présente XPointer[XPo02] pour sa plus grande flexibilité d'adressage dans chaque document. Spécialement pour des parties de document, comme des points ou intervalle qui ne serait pas défini à l'intérieur du même nœud parents. L'exemple pris est celui des espaces qu'il faut avoir entre deux liens. L'intervalle qui doit contenir du texte n'est pas facilement adressable avec les techniques XPath et XQuery.

- Avantage : Très puissante, cette technique permet d'exprimer des règles nettement plus complexes.
- Inconvénient : Pas aussi puissante que SQL pour la recherche d'information, ne contient pas de « Group by » par exemple.

- **Evaluation grâce à des règles basées sur des algorithmes complexes.**

Il est possible d'évaluer certaines règles d'ergonomie en utilisant des algorithmes complexes, comme c'est le cas de l'étude de David Chek Ling Ngo [Ngo00] qui développe une théorie mathématique pour l'esthétique d'une interface, ou de l'étude de Ravi Kothari,[Kot02] sur la proximité des éléments d'une page web mesuré selon la perception des éléments (couleur, densité, …).

- Avantage : Très puissante, les algorithmes complexes donnent des informations très intéressantes.
- Inconvénient : Très coûteuse à utiliser et implémenter.

- **Evaluation grâce à des règles exprimées en XSLT.**

L'étude ne parle pas de XSLT [Xsl99] comme technique d'évaluation, c'est cependant une technique assez utile. Elle est très proche de XPath et est principalement utilisée pour transformer le contenu d'un document Usixml par exemple, en un contenu différent. Ce langage est un langage de transformation. Cette technique est déjà implémentée dans l'outil d'édition graphique d'interface GrafiXML, pour l'exportation du document en langage XHTML, ou par le module en développement de « Graceful degradation » [Flo04]. Nous pourrions cependant l'utiliser pour générer un code XHTML contenant le résultat de l'analyse du document.

- Avantage : Aussi puissante que XPath, elle a le mérite d'être déjà implémentée dans l'outil GrafiXML.
- Inconvénient : C'est plus une technique de transformation, qu'une technique de requête comme XQuery.

Maintenant que nous avons vu les différentes techniques, nous pouvons envisager de traduire nos règles interprétées en des règles évaluables automatiquement.

6.2 Transformation des règles

Nous avons regardé différentes techniques possible pour l'évaluation des règles dans la section précédente. Nos règles interprétées pour Usixml peuvent maintenant être exprimées dans un langage plus technique grâce à l'une des techniques.

6.2.1 Quelques règles traduites

Nous avons fait le choix du langage XSLT pour plusieurs raisons, la première est qu'il est directement utilisable dans le logiciel GrafiXML, le seconde est que c'est un langage facile à prendre en main, tout en restant particulièrement puissant.

- *Les éléments VideoComponent, ImageComponent ont un attribut toolTipContent qui peut servir d'alternative textuelle. Cet attribut doit donc être renseigné pour tous les éléments graphiques.*

```
<xsl:template match="imageComponent | videoComponent [not (@toolTipContent)]">
        …
</xsl:template>
```

L'attribut toolTipContent est un attribut commun à tous les composants graphiques. Si d'autres éléments non textuels (pas image, ni vidéo) ont besoin d'une alternative textuelle, c'est cet attribut qui sera utilisé. On pourrait appliquer cette même règle mais de manière plus générale si nous avions en plus une méthode « *Enfant( )* ». Cette méthode remplacerait l'élément entre parenthèse par chacun des types d'éléments du modèle Concret d'Usixml qui hérite de cet élément

```
<xsl:template match="Enfant(graphicalIndividualComponents) [not (@toolTipContent)]">
        …
</xsl:template>
```

Même si les composants audio n'ont pas encore la possibilité d'avoir une alternative textuelle à cause de l'état actuel de développement d'Usixml. Si nous avions l'attribut toolTipContent, la règle s'exprimerait comme suit :

- *Les composants audio doivent avoir un attribut toolTipContent*

```
<xsl:template match="auditoryOutput[not (@toolTipContent)]">
        …
</xsl:template>
```

Avec la version 1.6.3 d'Usixml, nous pouvons vérifier que l'attribut toolTipContent ait une valeur non vide pour les éléments ayant IsMandatory mis à vrai, et une valeur vide pour tous les autres. IsMandatory serait utilisé pour indiquer que l'image est de décoration ou au contraire convoie de l'information importante.

```
<xsl:template match="imageComponent[not (@toolTipContent)]">
        …
</xsl:template>
<xsl:template match="imageComponent[@toolTipContent]">
        <xsl:choose>
        <xsl:when test='@isMandatory'>
                <xsl:if test='toolTipContent = “”'>
                        …
                </xsl:if>
        </xsl:when>
        <xsl:when test='not(@isMandatory)'>
                <xsl:if test='not(@toolTipContent = “”)'>
                        …
                </xsl:if>
        </xsl:when>
        </xsl:choose>
</xsl:template>
```

Une autre règle est, par exemple, celle concernant les éléments multimédia.

- *Les éléments multimédia videoComponent doivent avoir leurs attributs subtitle et subtitleContent.*

```
<xsl:template match="videoComponent[not (@subtitle)]">
        …
</xsl:template>
<xsl:template match="videoComponent[not (@subtitleContent)]">
        ….
</xsl:template>
```

Ces exemples sont assez simples, prenons maintenant un autre exemple, nous avons interprété une règle pour Usixml concernant la couleur comme suit :

- *Si une partie de texte change de couleur (textColor, bgColor ou fgColor) par rapport au reste du texte, alors il faut que le style de cette partie de texte change aussi (isBold, isItalic, textFont, etc.)*

Supposons que nous avons le texte suivant :

> Le chancelier suprême dit : « Il est grand temps que nos soldats attaquent. **L'ennemi** n'est sûrement pas encore prêt à se défendre ».

Ce texte ci-dessus écrit principalement en couleur noire, contient deux parties en couleur ; « chancelier suprême » est en orange (#FF9900) et « L'ennemi » est en couleur rouge (#FF0000). Cependant, sur un écran monochrome, nous ne pourrions pas voir la couleur et le texte apparaîtrait dans une même couleur (ici en noir).

> Le chancelier suprême dit : « Il est grand temps que nos soldats attaquent. **L'ennemi** n'est sûrement pas encore prêt à se défendre ».

La mise en évidence du texte « chancelier suprême » ne sera pas visible, alors que le texte « l'ennemi » le sera bien, puisqu'en plus d'être de couleur rouge, il est mis en **gras**. Voyons maintenant le code Usixml où nous avons supprimé les informations inutiles pour ne pas surcharger l'exemple.

Source

```
1       <box >
2               <textComponent id="text_component_1" name="text_component_1"
3                       defaultContent="Le "
4                       isBold="false"
5                       textColor="#000000"
6               />
7               <textComponent id="text_component_2" name="text_component_2"
8                       defaultContent="Chancelier suprême"
9                       isBold="false"
10                      textColor=" # FF9900"
11              />
12              <textComponent id="text_component_3" name="text_component_3"
13                      defaultContent=" a dit : &quot; Il est grand temps que nos
soldats attaquent. "
14                      isBold="false"
15                      textColor="#000000"
16              />
17              <textComponent id="text_component_4" name="text_component_4"
18                      defaultContent="L'ennemi"
19                      isBold="true"
20                      textColor="#FF0000"
21              />
22              <textComponent id="text_component_5" name="text_component_5"
```

```
23                      defaultContent=" n'est sûrement pas encore prêt à se défendre
». "
24                      isBold="false"
25                      textColor="#000000"
26              />
27      </box>
```

Pour illustrer l'exemple, nous nous sommes concentrés sur l'attribut isBold comme unique critère de style.

XSLT stylesheet

```
1       <xsl:stylesheet version = '1.0'
2       xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
3
4       <xsl:variable name="voisin">
5               <xsl:value-of select=0/>
6       </xsl:variable>
7
8       <xsl:template match="//textComponent">
9
10              <xsl:variable name="text_color">
11                      <xsl:value-of select="./@textColor"/>
12              </xsl:variable>
13              <xsl:variable name="Bold">
14                      <xsl:value-of select="./@isBold"/>
15              </xsl:variable>
16              <xsl:variable name="position">
17                      <xsl:value-of select="position()"/>
18              </xsl:variable>
19              <xsl: if test = “position() != 1”>
20                      <xsl:for-each select="preceding-sibling::*">
21                      <xsl:if test="position()+1 = $position">
22                              <xsl:if test="@textColor != $text_Color">
23                                      <xsl:if test="@isBold = $Bold">
24                                              <xsl:variable name="voisin">
25                                              <xsl:value-of select= $voisin +1/>
26                                              </xsl:variable>
27                                      </xsl:if>
28                              </xsl:if>
29                      </xsl:if>
30                      </xsl:for-each>
31              </xsl: if>
32              <xsl: if test = “position() != last()”>
33                      <xsl:for-each select="following-sibling::*">
34                      <xsl:if test="position()-1 = $position">
35                              <xsl:if test="@textColor != $text_Color">
36                                      <xsl:if test="@isBold = $Bold">
37                                              <xsl:variable name="voisin">
38                                              <xsl:value-of select= $voisin +1/>
39                                              </xsl:variable>
40                                      </xsl:if>
41                              </xsl:if>
42                      </xsl:if>
43                      </xsl:for-each>
```

```
44                </xsl: if>
45                <xsl:choose>
46                <xsl:when test="(position() = 1) or (position() = last())">
47                        <xsl:if test="$voisin = 1">
48                                <div style="color:red">
49                                <xsl:text>Erreur 5 : </xsl:text>
50                                </div>
51                                <xsl:text>Composant de type :</xsl:text>
52                                <xsl:value-of select="name()"/>
53                                <xsl:text> [Id = </xsl:text>
54                                <xsl:value-of select="@id"/>
55                                <xsl:text> ]</br></xsl:text>
56                        </xsl:if>
57                </xsl:when>
58                <xsl:otherwise>
59                        <xsl:if test="$voisin = 2">
60                                <div style="color:red">
61                                <xsl:text>Erreur 5 : </xsl:text>
62                                </div>
63                                <xsl:text>Composant de type :</xsl:text>
64                                l:value-of select="name()"/>
65                                <xsl:text> [Id = </xsl:text>
66                                <xsl:value-of select="@id"/>
67                                <xsl:text> ]</br></xsl:text>
68                        </xsl:if>
69                </xsl:otherwise>
70                </xsl:choose>
71        </xsl:template>
```

Output

```
<div style="color:red">Erreur 5 : </div>
Composant de type :
textComponent
[Id =
text_component_1
 ]</br>
<div style="color:red">Erreur 5 : </div>
Composant de type :
textComponent
[Id =
text_component_2
 ]</br>
```

HTML view

Erreur 5 : Composant de type : textComponent [Id = text_component_1]
Erreur 5 : Composant de type : textComponent [Id = text_component_2]

Le résultat nous indique que le premier et le deuxième composant texte ne sont pas bien formés, ils ne correspondent pas à la règle émise, puisqu'il change de couleur alors qu'il n'y a pas d'autres changements de style (dans notre cas, il s'agit de la mise en gras). Par contre, le quatrième composant texte vérifie correctement la règle.

Notons que nous prenons le premier composant texte également car il est de couleurs différentes au composant texte suivant, sans changer de style, et il n'est pas censé savoir que la couleur de base est le noir et non l'orange. C'est toute la difficulté des débuts et fins de texte.

6.2.2 Limitations des techniques choisies pour l'implémentation

Comme il est dit dans l'article de Vicente Luque Centeno [Cen05], il n'est pas toujours possible d'évaluer les règles, car nous n'avons pas les outils adéquats ou que l'algorithme nécessaire serait trop complexe.

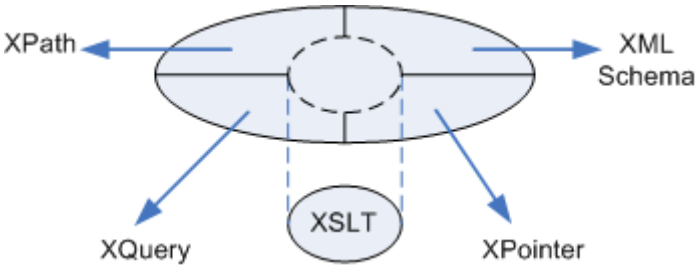


Figure 20 - restriction des règles selon la technique utilisée

Nous avons utilisé le langage XSLT comme technique pour évaluer nos règles. Et comme le montre la Figure 20, nous ne pouvons pas utiliser le langage XSLT pour couvrir toutes les règles. Tout comme le langage, la technique a des limitations qui restreignent encore notre ensemble de règles.

Evidement, il est possible d'évaluer les règles avec d'autres techniques comme nous le présentons dans l'exemple suivant avec le langage XQuery.

- *Fournir des liens redondants pour les zones d'image*

```
//imageZone
[
let $imageZone:=self::imageZone
return
count(//textComponent[@hyperLinkTarget = $imageZone/@hyperLinkTarget])= 0
]
```

Pour couvrir efficacement le plus de règles possibles, il est intéressant de combiner les différentes techniques, si cela est rendu possible.

6.3 Différentes suggestions d'Intégration à un outil existant.

Parmi les outils qui utilisent le langage Usixml, deux éditeurs graphiques semblent particulièrement intéressants pour intégrer une validation de règle d'accessibilité et d'ergonomie. Il s'agit des outils GrafiXML et SketchiXML [Coy04] [Coy05]. Nous présentons dans les sous-sections suivantes, les différentes possibilités d'intégration pour chacun des outils.

6.3.1 GrafiXML

- **Présentation**

GrafiXML est un outil graphique qui permet de dessiner des interfaces graphiques, en ajoutant des fenêtres, des widgets, des composants images dans une surface représentant l'interface, de manière similaire à Dreamweaver pour les interfaces web (voir la Figure 21)

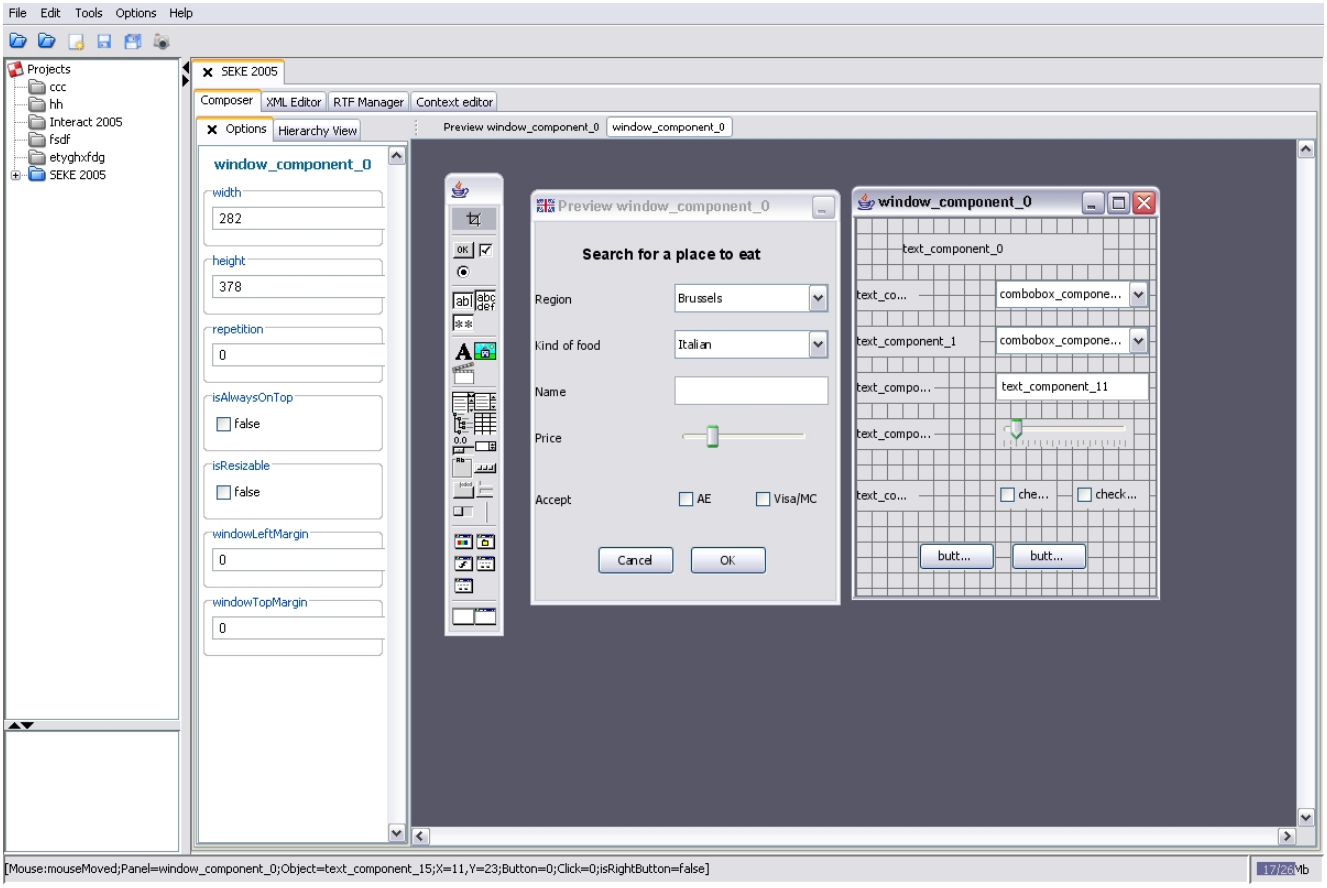


Figure 21 - Le logiciel GrafiXML.

Cet outil se base sur le langage Usixml comme représentation de l'interface graphique. Plus exactement, le document Usixml est généré en même temps que l'utilisateur insert, supprime ou modifie des éléments de l'interface. Ce document peut être consulté à tout moment.

Puisque nos règles s'appliquent au document de type Usixml et que GrafiXML produit un document de ce type qui est, à tout moment, consultable. Il serait intéressant de permettre à l'utilisateur de vérifier que l'interface, qu'il a créée, respecte certaines règles d'accessibilité telles que celles que nous avons identifiées.

L'ajout d'un plug-in, c'est-à-dire d'une fonctionnalité supplémentaire au logiciel, qui serait en charge de l'évaluation des règles exprimées dans une des techniques précitées, permettrait à l'utilisateur de demander au logiciel d'évaluer le document au même titre que LIFT pour Dreamweaver [Lif05].

- **Limitation dans l'évaluation des règles.**

GrafiXml est un éditeur graphique, il prend en compte le modèle concret d'Usixml, mais pas le modèle des tâches, ni le modèle du domaine, etc. L'ensemble de nos règles ne sont donc pas tous évaluables via un plug-in à GrafiXML. D'autre part, il est possible que certaines techniques d'évaluation ne soient pas prises en compte dans GrafiXML. En effet, suite à une entrevue avec Benjamin Michotte, responsable de GrafiXML, nous avons appris que la validation du document Usixml par un schéma XML particulier n'est pas encore implémentée, mais le sera prochainement.

- **Suggestion d'utilisation**

Lorsque l'utilisateur veut évaluer son document, son interface fraîchement créée, il pourra demander au plug-in de vérifier le document pour différentes règles d'accessibilité et d'ergonomie. On devrait pouvoir donner à l'utilisateur de choisir les règles qu'il désire évaluer. Un système de sélection des règles pourrait être le suivant :
Des règles générales regroupant des règles plus spécifiques, seraient classifiées selon la classification du chapitre 4, et des règles plus spécifiques classifiées selon leurs champs d'action, c'est-à-dire par éléments.

L'utilisateur sélectionne les règles qu'il désire et lance l'évaluation. Le résultat serait une liste d'erreurs indiquant et expliquant quelle règle n'est pas vérifiée, de manière similaire à WebXACT par exemple.

Mais mieux encore, nous pourrions donner l'occasion à l'utilisateur de demander que son document soit vérifié tout le temps. En effet, GrafiXML a l'avantage de générer en temps réel le document Usixml, cela permettrait également au plug-in d'agir en temps réel. Si l'utilisateur veut une évaluation temps réel, il lui suffit de configurer les règles qu'il désire évaluer dans la liste, de cocher la validation temps réel.

Prenons l'exemple de l'attribut toolTipContent qui doit être renseigné pour toutes les images. Lorsque l'utilisateur ajoute une image au document, notre plug-in en charge de l'évaluation indiquerait automatiquement à l'utilisateur que l'action qu'il vient d'effectuer n'est pas conforme aux règles d'accessibilité dans une fenêtre supplémentaire. Le message indiquerait que le composant image qu'il vient d'ajouter doit contenir une valeur pour l'attribut toolTipContent non vide. A ce moment, l'utilisateur peut soit introduire le contenu de l'attribut et la règle est validée, soit il peut indiquer que cette image ne doit pas avoir de contenu non vide pour cette image parce que c'est une image

de décoration, et dans ce cas indiquer l'attribut isMandatory à faux, soit encore ignorer l'erreur. Dans les deux premières possibilités, le message est supprimé puisque le document est à nouveau valide. Dans la dernière possibilité, le message reste présent et signale toujours à l'utilisateur que son document n'est pas valide pour ce type de règles.

- **Suggestion d'implémentation**

Actuellement, il existe un plug-in encore en développement concernant la « Graceful Degradation », Ce plug-in se base sur XSLT pour produire une interface joliment dégradée plus adaptée à différentes plates-formes.

Nous pourrions très bien utiliser la base de ce plug-in pour développer l'outil d'évaluation. En effet, avec le langage XSLT nous pourrions vérifier certaines règles et produire un document XML contenant l'erreur et l'explication de l'erreur pour chacune des règles trouvées.

En effet, le code suivant vérifie que toutes les images ont un texte alternatif, et renvoie une partie de document XML.

XSLT stylesheet

```
<xsl:stylesheet version = '1.0'
    xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>

<xsl:template match="imageComponent[not (@toolTipContent)]">
    <div style="color:red">
    <xsl:text>Erreur 45 : </xsl:text>
    </div>
    <xsl:text>Composant de type : </xsl:text>
    <xsl:value-of select="name()"/>
    <xsl:text> [Id = </xsl:text>
    <xsl:value-of select="@id"/>
    <xsl:text> ]</xsl:text>
</xsl:template>

</xsl:stylesheet><
```

Output

```
<div style="color:red">Erreur 45 : </div>
Composant de type :
imageComponent
 [Id =
4
 ]
```

HTML view

Erreur 45 : Composant de type : imageComponent [Id = 4]

6.3.2 SketchiXML

- **Présentation**

SketchiXML est un outil d'édition graphique multi-agent pour la conception des interfaces homme-machine. Cet outil est basé sur la reconnaissance de dessins faits à la main sur une surface tactile. Comme nous le montre la Figure 22 et la Figure 23 , il y a une correspondance entre des dessins et des widgets ou autres composants graphiques, ainsi un rond suivi d'une ligne est interprété comme un radio bouton, par exemple.

SketchiXML est destiné à aider un ensemble de personnes qui ne sont pas développeurs à créer un prototype à basse fidélité d'une interface graphique [Coy05]. L'avantage de SketchiXML est qu'il permet de dessiner touts interface de façon similaire à celle que l'on fait avec un papier et un crayon, avec la possibilité d'effacer, de déplacer, de créer. C'est-à-dire que l'on ne perd pas la flexibilité si précieuse au papier et au crayon. Mais SketchiXML va plus loin puisqu'il assiste l'utilisateur par la reconnaissance des formes, mais aussi par la génération d'un document Usixml décrivant l'interface dessinée.

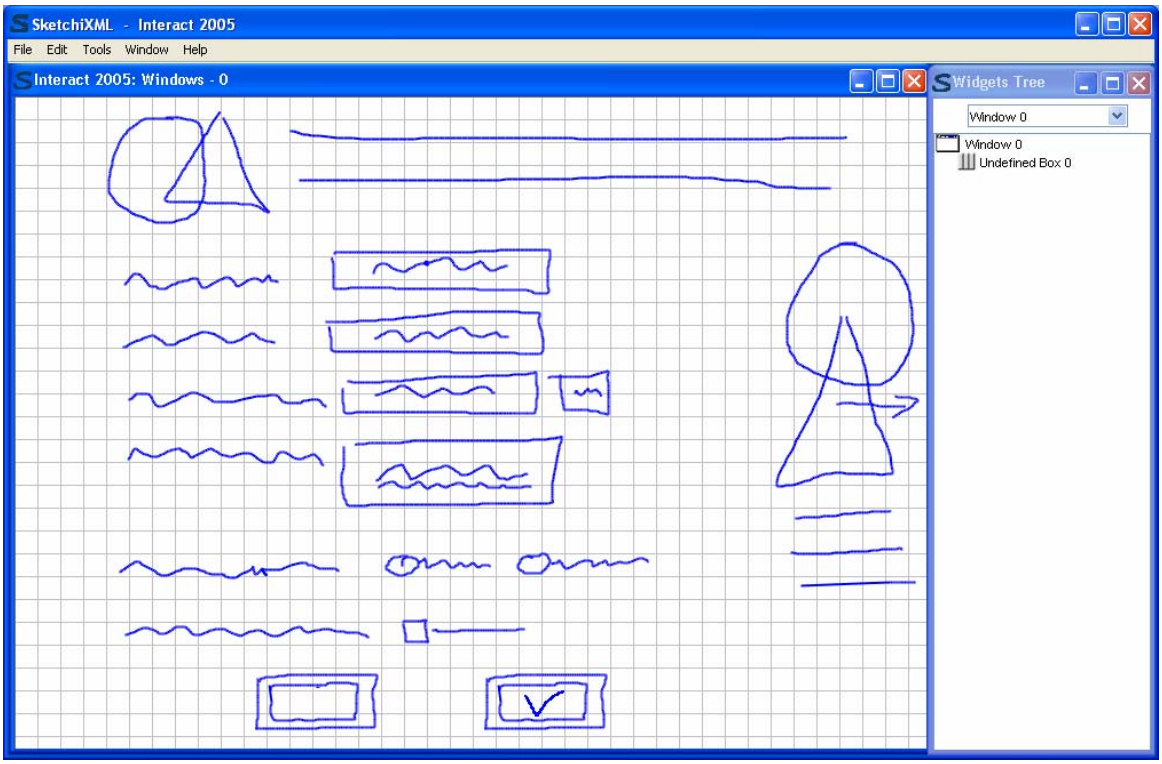


Figure 22 - Ecran de SketchiXML où l'utilisateur à dessiner à la main les composants

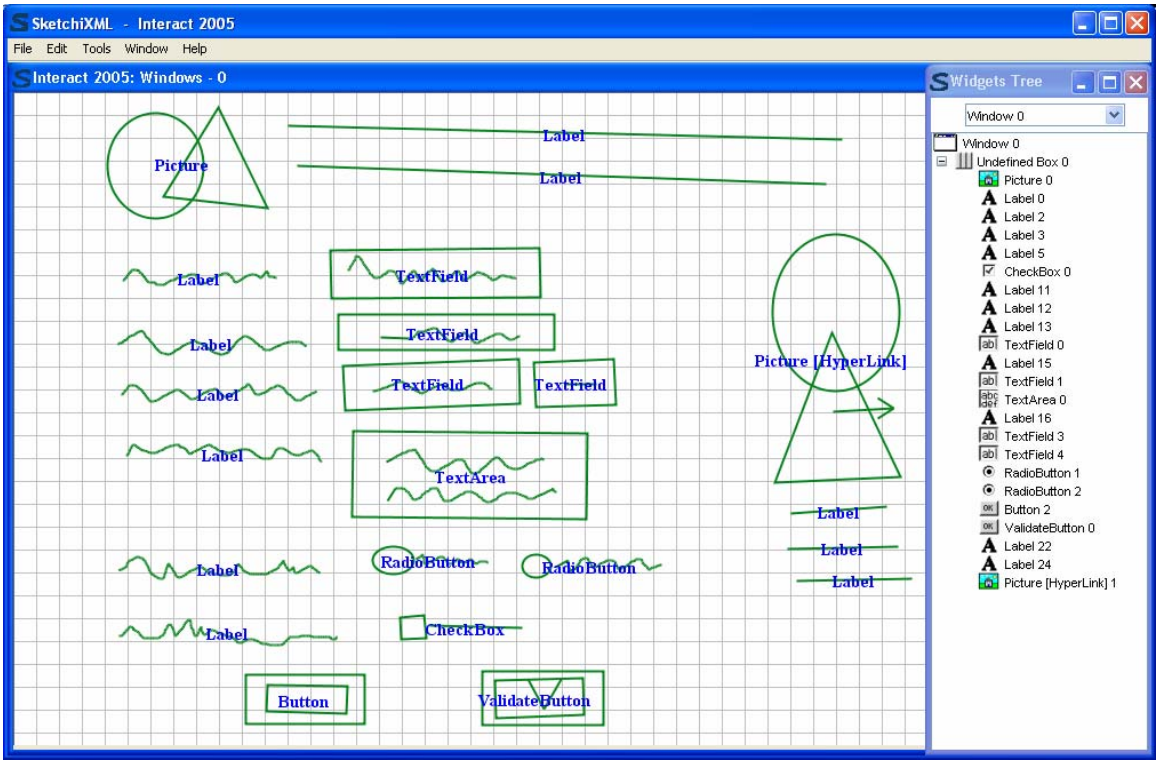


Figure 23 - Ecran de SketchiXml où le logiciel a reconnu les composants dessinés.

- **Limitation**

Tout comme pour GrafiXML, SketchiXML ne prend en compte que le modèle concret du langage Usixml. Mais, contrairement à GrafiXML qui correspond plus à un outil de haute fidélité, dans le sens où l'interface réalisée avec GrafiXML est très fidèle au résultat final attendu. SketchiXML se restreint encore plus à la présentation de l'interface, c'est un outil de basse fidélité, il ne se soucie pas du contenu des textComponent ou des checkbox, ni de la typographie, ni des couleurs. Cette limitation nous restreint fortement dans les règles d'ergonomie et d'accessibilité applicable à cet outil. Pour revenir avec notre exemple classique d'alternative textuelle, cette règle n'as aucun intérêt dans cet outil, puisque ce dernier ne se soucie pas de ce type de règle. Par contre, des règles d'ergonomie comme l'équilibre d'une interface, la symétrie, la constance de la présentation sont des règles applicables, comme les règles suivantes issues de « The Essential Guide of User Interface Design » de Galitz [Gal96].

Group Boxes – Proper usage

- *To provide a border around radio button or check box controls*
- *To provide a border around two or more functionally related controls*

- **Première suggestion d'utilisation**

L'utilisation d'un outil d'évaluation de règles intégrées à SketchiXML est assez semblable à celle imaginée pour GrafiXML. A savoir, un plug-in proposant à l'utilisateur une liste de règles qu'il peut sélectionner et ensuite exécuter l'évaluation. Le résultat serait constitué d'une liste d'erreurs indiquant le ou les objets en erreur.

Un inconvénient que possède SketchiXML par rapport à GrafiXML et qui limite les possibilités actuelles d'intégration est qu'il ne génère pas en temps réel le document Usixml correspondant. C'est-à-dire que lorsque l'utilisateur demandera une évaluation ergonomique en cours de conception, on sera obligé de générer le code Usixml de manière temporaire, d'évaluer les règles et d'informer l'utilisateur des erreurs. Le plus difficile sera de pouvoir faire la correspondance, après l'évaluation, entre les erreurs observées et le prototype illustré à l'écran.

SketchiXML gagnerait à générer en temps réel le document Usixml afin d'avoir une fonctionnalité similaire à celle suggérée pour GrafiXML. L'ajout d'un outil d'évaluation continue pourrait interrompre les utilisateurs pour suggérer une solution plus évidente, plus ergonomique. Par exemple, si l'on observe que l'utilisateur dessine plus de 9 radios bouton formant un ensemble, il est préférable d'utiliser une listbox fixe, ou une liste drop-down listbox si l'espace est restreint.

- **Seconde suggestion d'utilisation**

Une seconde utilisation des règles d'ergonomie pour l'outil SketchiXML, serait de pouvoir arranger les dessins que l'utilisateur crée sur l'écran. Dans un premier temps, l'utilisateur devrait configurer l'outil en indiquant les règles qu'il souhaite voir appliquer. Ensuite, après qu'il ait dessiné un ensemble de widget particulier, il aurait l'occasion de demander de les arranger pour obtenir une interface plus ergonomique. L'arrangement ne modifierait que peu l'apparence donnée aux dessins, mais les rendraient homogènes, alignés selon les règles d'ergonomie sélectionnées.

Voici un exemple de règle interprétable pour Usixml et qui conviendrait à SketchiXML.

> *Horizontal Orientation and Vertical Alignement*
> *Radio Buttons/ Check Boxes Selection Controls*
> - *Align leftmost radio buttons and/or check boxes.*
> - *Field Captions may be left- or right-aligned.*

Chapitre 7 Conclusion

Il n'est pas facile de s'arrêter dans une étude comme celle que nous avons entreprise. Nous avons entièrement présenté notre méthodologie et il est temps maintenant de conclure.

Dans la section 7.1, nous présenterons les avantages et inconvénients de l'étude et nous terminerons par la section 7.2, où nous parlerons des possibilités pour des travaux futurs.

7.1 Avantages / Inconvénients

Cette étude comporte des avantages et des inconvénients. En effet, l'un des inconvénients est celui que nous avons déjà énoncé lors de la présentation de notre méthodologie, à savoir le fait, que l'ensemble des règles se restreint indéniablement par nos choix technologiques. Si nous avions choisi d'autres critères de classification, un autre langage que celui d'Usixml ou une autre technique d'implémentation, notre ensemble de règles serait peut-être plus large, mais pourrait être, probablement, plus étroit également.

Par contre, l'avantage de cette étude vient de notre méthodologie, à savoir que si l'on veut utiliser une autre technique d'implémentation que celle que nous avons choisie, on peut repartir du résultat obtenu à la troisième étape de notre méthodologie. Si l'on veut choisir un autre langage, comme par exemple XUL, nous pouvons garder la classification des règles d'accessibilité que nous avons établies (deuxième étape de notre méthodologie).

Un autre inconvénient de cette étude est le choix du langage. Loin de nous, l'idée de remettre en cause ce choix qui n'est point mauvais. En réalité, comme nous l'avons déjà présenté au chapitre 5, l'inconvénient inhérent au langage d'Usixml est que ce dernier n'est pas encore en développement. Il est certain que ce langage continuera d'être développé et c'est très bien ainsi, nous l'encourageons par ailleurs. L'inconvénient est que cette étude s'est basée sur la documentation 1.4.3, ainsi que sur la version 1.6.1. Au cours de l'élaboration de cette étude, la version 1.6.3 est sortie. Cela ne devrait normalement pas poser de problème si la version 1.6.3 ne faisait qu'ajouter des composants supplémentaires qui sont d'ailleurs particulièrement intéressants tel que l'attribut isMandatory dont nous avons largement parlé. En effet, cette version effectue également la modification suivante : les attributs textFont, isBold, isItalic, isUderline, isStrike, isSubScript, isSuperScript, isPreformated, textSize, textColor sont déplacés vers le Graphical Concrete Interaction Object component, c'est-à-dire vers un élément parent. Dans un sens, cela ne change pas les règles déjà interprétées puisque chacun des éléments qui contenait ces attributs, les héritent maintenant de leurs parents. Mais cela change la donne, dans les façons d'interpréter les règles, au point que certaines règles seront peut-être revues et d'autres règles devront être interprétées.

Dans un certain sens, les règles que nous avons justement considérées comme non applicables à Usixml à cause de son état actuel de développement, devront à chaque nouvelle version d'Usixml être réexaminée. Celles-ci pourront, le cas échéant, être interprétées et rendues applicables à Usixml.

L'étude contient aussi des avantages. L'un des avantages, et cela peut sembler contradictoire avec l'inconvénient que nous venons de révéler, est l'usage du langage Usixml. L'avantage d'Usixml est qu'il couvre l'interface homme-machine en général, qu'il intègre beaucoup d'informations concernant la description des interfaces hommes-machines. Là où apparaît l'avantage de cette étude, et plus précisément l'interprétation et l'évaluation des règles pour ce langage, c'est qu'il existe déjà des transformations de ce langage vers d'autres langages plus spécifiques comme QTk par l'interpréteur QTkiXML,

Java, HTML, XUL, PDF, et d'autres à venir. L'avantage donc de s'attaquer à l'évaluation de règles pour le langage Usixml revient à le faire pour les autres langages qui peuvent en découler. Par ailleurs, pour des langages comme XUL, HTML, et d'autres langages basés sur XML, il nous semble qu'il serait possible d'adapter les règles de transformations du document Usixml vers le document source, pour que ces règles effectuent également la transformation des règles d'ergonomie interprétées pour Usixml en des règles interprétées pour le langage de destination. Ainsi le travail effectué pour Usixml pourrait servir de base pour le travail à effectuer si l'on désire interpréter les règles pour un langage basé sur XML, tel que XUL par exemple.

Par exemple la règle qui dit que tous les éléments de menu doivent avoir une mnémonique …

- s'exprimera en XSLT pour Usixml comme suit :

```
<xsl:template match="menuItem[not (@mnemonic)]">
</xsl:template>
```

- s'exprimera en XSLT pour XUL comme suit :

```
<xsl:template match="menuitem[not (@mnemonic)]">
</xsl:template>
```

- s'exprimera en XSLT pour XAML comme suit :

```
 <xsl:template match="MenuItem[not (@mnemonic)]">
</xsl:template>
```

L'exemple est évidemment choisi, mais on se rend compte que, si les langages gardent une structure assez similaire, certaines règles d'ergonomie peuvent très facilement se traduire dans l'un des autres langages.

7.2 Travaux futurs

Évidemment, l'étude que nous avons réalisée, peut servir de point de départ pour certains travaux futurs.

Nous avons décidé de présenter les différentes possibilités de travaux futurs à chaque étape de notre méthodologie.

7.2.1 Recherche et Collation

Concernant la recherche et la collation, il est toujours possible d'étendre la recherche à des règles d'ergonomie et d'accessibilité, mais plus encore à des règles peut-être plus spécifiques. En effet, nous avons des règles spécifiques pour les environnements Windows, Apple, Sun, et autres. Certaines de celles-ci ne sont pas, à proprement parler, des règles d'ergonomie dans le sens strict du terme, mais constituent un « plus » pour une conformité aux « look and feel » des différents systèmes. D'autres règles encore, sont celles qui dépendent de la plate-forme de destination, c'est-à-dire des règles de mise en

page propres aux PDA, aux écrans muraux, aux ordinateurs de bureau ou aux ordinateurs portables. Ces règles peuvent très bien, également, être ajoutées à celles que nous avons prises en compte, la démarche d'interprétation et d'implémentation restant la même.

Outre des règles d'ergonomie, nous pouvons également rechercher d'autres règles d'ergonomie plus générales ou des propriétés d'interface. Nous pensons, par exemple, aux travaux de l'équipe IIHM du laboratoire CLIPS (Communication Langagière et Interaction Personne-Système) [Cli05] en collaboration avec les membres du groupe de travail WG2.7 de l'IFIP (International Federation for Information Processing) [Ifi05] qui présente l'utilisabilité d'une interface comme l'addition des principes de **Souplesse** (qui exprime l'éventail des choix pour l'utilisateur et le système) et de **Robustesse** (qui vise la prévention des erreurs et l'augmentation des chances de succès de l'utilisateur). [Cou05] Sous le principe de souplesse, il exprime des propriétés tel que l'atteignabilité, la non préemption, l'interaction multifilaire, la multiplicité de rendu, l'adaptativité, la plasticité, la migrabilité de tâches, par exemple. Sous le principe de robustesse, il exprime les propriétés telles que l'observabilité, la réciprocité, la réflexivité, l'honnêteté, la viscosité, par exemple. C'est dire combien il existe d'autres propriétés qui peuvent être prises en compte dans l'évaluation ergonomique. Notons cependant que l'avantage de notre étude est de montrer une méthodologie qui peut s'appliquer à toute étude similaire, quelle que soient les règles prises en compte. En effet, la méthodologie que nous avons présentée reste la même, même si ces propriétés sont terriblement abstraites, comme l'honnêteté par exemple, et que ces propriétés ne sont certainement pas facilement interprétables ou prendraient énormément de temps pour l'être à 100%,

7.2.2 Classification

Au niveau de la classification, d'autres critères de classification sont possibles. Plus exactement, il serait intéressant de classifier les règles interprétées pour un langage particulier. En effet, nous nous rendons compte qu'une classification supplémentaire après l'interprétation permettrait de savoir plus facilement quelles sont les règles qu'il est possible d'évaluer.

Une des améliorations que l'on peut faire à l'étude, ici présente, serait de classifier les règles interprétées selon les modèles nécessaires à leur implémentation. Pour chacune des règles, nous avons déjà indiqué quels sont les modèles nécessaires. L'avantage d'une classification selon ce critère nous donne une meilleure idée de l'ensemble des règles applicables à un outil qui ne prend en compte qu'un ou plusieurs des modèles d'Usixml. Les exemples de GrafiXML ou de SketchiXML que nous avons présentés dans le chapitre précédent montre bien la limite d'application de notre étude à certains outils.

Une autre classification possible est une classification par widgets, par éléments, c'est-à-dire de classifier les règles selon leurs champs d'application. Par exemple, regrouper toutes les règles qui s'appliquent aux textComponent.

7.2.3 Interprétation

Outre des interprétations pour des langages tel que XAML ou XUL que nous avons présentés, dans la section précédente, il est également possible d'interpréter les règles pour des langages comme UIML ou XIML pour ne parler que de ceux là.
Ces langages qui sont compatibles XML présentent des avantages similaires à Usixml. UIML (User Interface Markup Language) [Abr99] est également un langage de description d'interface homme-machine. Il permet au concepteur de décrire l'interface utilisateur en des termes générique et d'utiliser une description de style pour faire correspondre l'interface utilisateur à différents systèmes d'exploitation, langages ou outils. XIML (eXtensible Interface Markup Language) fournit, lui, un moyen de définir une interface homme-machine sans se soucier de l'implémentation. Son but est de décrire les aspects abstraits de l'interface utilisateur, c'est-à-dire les tâches, l'utilisateur et le domaine, ainsi que les aspects concrets, c'est-à-dire la présentation et le dialogue.

Ces deux langages sont des langages compatibles XML. Ils couvrent des aspects similaires à ceux couverts par Usixml, les aspects tâches, domaine, utilisateur, concret et abstrait. L'interprétation des règles d'ergonomie et d'accessibilité pour ces langages constitue une continuité dans les deux premières étapes de notre méthodologie. En effet, seul le choix du langage particulier serait différent.

Pour revenir à notre étude, nous avons principalement dirigé notre étude vers l'interprétation des règles d'accessibilité. L'ensemble des règles d'ergonomie constitue un domaine qu'il est intéressant d'approfondir et d'interpréter pour le langage Usixml.

7.2.4 Evaluation et implémentation

Dans le chapitre 6 nous avons présenté deux suggestions d'utilisateur de cette étude dans des outils d'édition graphique existants. La mise en œuvre de ces suggestions constitue des travaux futurs potentiels. Nous avons déjà un petit peu détaillé les techniques possibles permettant de réaliser ces intégrations.

D'autres travaux sont également possibles, nous pensons, par exemple, à un outil d'évaluation indépendant des autres outils existants. C'est-à-dire à un outil qui permettrait d'évaluer tous documents du type Usixml incluant ou non tous les modèles. En effet, les suggestions d'intégration dans les outils existants tel que GrafiXML et SketchiXML limitent la portée d'évaluation des règles au seul modèle concret d'Usixml, alors qu'il est également possible d'utiliser les autres modèles pour tester d'autres règles ou affiner d'autres règles encore.

Chapitre 8 Bibliographie

0-9

[508] CITA (Center for Information Technology Accommodation), *U.S. Section 508 Guidelines*,1998. Accessible à http://www.section508.gov/index.cfm

A

[Abr99] ABRAMS, M., PHANOURIOU, C., BATONGBACAL, SHUSTER, J., *UIML: An Appliance-Independent XML User Interface Language*, Proc. of 8th World Wide Web Conference WWW'8 ,Toronto, Mai 1999.

[Acc03] ACCESSIWEB, *Accessibilité du Web – Centre de ressources et de recherche sur l'accessibilité du Web*, 2003, consulté le 02-08-2005. Accessible à http://www.accessiweb.org/

[Ada90] U.S. DEPARTMENT OF JUSTICE, *Americans with Disabilities Act*, 1990. Accessible à http://www.usdoj.gov/crt/ada/adahom1.htm

[App87] APPLE, *Human Interface Guidelines: the Apple desktop interface*. CA: Addison- Wesley, 1987.

[Apr02] ATRC, *A-Prompt : Web Accessibility Verifier*, Adaptive Technology Resource Centre (University of Toronto) and Trace Center (University of Wisconsin), Canada & Usa, Avril 2002. Accessible à http://aprompt.snow.utoronto.ca/

[Att05] ATTERER, R., SCHMIDT, A., *Adding Usability to Web Engineering Models and Tools*. Proceedings of the 5th International Conference on Web Engineering, Sydney, Australia, Juillet 2005.

B

[Bas93] BASTIEN, J.M.C., SCAPIN, D.L., *Ergonomic Criteria for the Evaluation of Human-Computer Interfaces*, Technical Report 156, INRIA, Rocquencourt, 1993.

[Bas98] BASTIEN, J.M.C., LEULIER, C., SCAPIN, D.L., *L'ergonomie des sites web*, INRIA Rocquencourt, 1998, pages 111-173. Accessible à http://www.adbs.fr/uploads/ouvrages/inria98/p111-173.pdf

[Bli03] BLINDSURFER, *Rudi Canters, fondateur et père spirituel du projet BlindSurfer est décédé inopinément le 4 avril 2003*, 2003. Accessible à htttp://www.blindsurfer.be/bsrudiF.htm

[Bli05] BLINDSURFER, consulté le 15-07-2005. Accessible à http://www.blindsurfer.be/bsindexF.htm

[Bra05a] BRAJNIK, G., CANCILA, D., NICOLI, D., PIGNATELLI, M., *Do text transcoders improve usability for disabled users?*, International Cross-Disciplinary Workshop on Web Accessibility, Mai 2005, Chiba, Japan.

[Bra05b] BRAJNIK, G., CANCILA, D., NICOLI, D., PIGNATELLI, M., *Do dynamic text-only pages improve usability for PDA users ?*, Proceedings of the 5th International Conference on Web Engineering, Sydney, Australia, Juillet 2005.

[BrN05] ASSOCIATION BRAILLENET, consulté le 15-07-2005. Accessible à http://www.braillenet.org/

[Bro88] BROWN, C.M., *Human-Computer Interface Design Guidelines*. Ablex Publishing Corp., Berkeley, 1988.

C

[Cam05] CAMELEON PROJECT : *Plasticity of user interfaces*, consulté le 12-08-2005. Accessible à http://giove.cnuce.cnr.it/cameleon.html

[Cas05] CAST, *Center for Applied Special Technology*, consulté le 21-08-2005. Accessible à http://www.cast.org/

[Cen05] CENTENO, V. L., KLOOS, C. D., GAEDKE, M., NUSSBAUMER, M., *WCAG Fomalization with W3C Techniques*, Proceedings of the 5th International Conference on Web Engineering, Sydney, Australia, 2005.

[Cli05] LABORATOIRE CLIPS, *Communication Langagière et Interaction Personne-Système*, 2005 accessible à http://www-clips.imag.fr/

[Con00] CONNELL, I.W., *The Use of Cognitive and Other Principles in Usability Evaluation: principles versus heuristics and cumulative problem curves*. Doctoral thesis, University of York Department of Psychology, October 2000.

[Cop95] COPE, M.E. and ULIANO, K.C., *Cost-justifying usability engineering: A real world example*, Proceedings of the Human Factors and Ergonomics Society 39th Annual Meeting, 1995.

[Cor05] CORMIER, B., PC Inpact, *Un clavier totalement revu pour l'ergonomie, 495 dollars*,consulté le 03-08-2005. Accessible à http://www.pcinpact.com/actu/news/Un_clavier_totalement_revu_pour_lergonomie_495_dol.htm

[Cou05] COUTAZ, J., NIGAY, L., *Propriétés en Interaction Homme-Machine*, consulté le 20-08-2005. Accessible à http://iihm.imag.fr/coutaz.book/coutaz.properties.html

[Coy04] COYETTE, A., FAULKNER, S., KOLP, M., LIMBOURG, Q., VANDERDONCKT, J., SketchiXML: *Towards a Multi-Agent Design Tool for Sketching User Interfaces Based on UsiXML*, Proc. of 3rd Int. Workshop on Task Models and Diagrams for user interface design TAMODIA'2004 (Prague, November 15-16, 2004), Ph. Palanque, P. Slavik, M. Winckler (eds.), ACM Press, New York, 2004.

[Coy05] COYETTE, A., VANDERDONCKT, J., *A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces*. Sera présenté à l'Interact 2005 : Communicating Naturally through Computers, 12-16 septembre 2005, Rome, Italie.

[CSG05] WWW.PIXY.CS, *Color Scheme Generator 2*, consulté le 15-08-2005. Accessible à http://wellstyled.com/tools/colorscheme2/index-en.html

D

[Del03] DELORIE, M., *Lynx Viewer*, 2003, consulté le 15-8-2005. Accessible à http://www.delorie.com/web/lynxview.html

[Den05] DENIS, V., *Un pas vers le poste de travail unique : QTKIXML un interpréteur d'interface utilisateur a partir de sa description*. Mémoire-Projet, 2005.

[Dic04] DICKEY, T., *Lynx 2-8-5*, 2004, consulté le 09-08-2005. Accessible à http://lynx.browser.org/

F

[Far96] FARENC, C., LIBERATI, V., BARTHET, M.-F., *Automatic Ergonomic Evaluation : What are the Limits ?*, in Proceedings of CADUI'96, 2nd International Workshop on Computer-Aided Design of User Interfaces - Namur, Belgique, 5-7 juin 1996.

[Flo04] FLORINS, M., VANDERDONCKT, J.,*Graceful degradation of user interfaces as a design method for multiplatform systems*. In: Nunes, Nuno Jardim, Rich, Charles (ed.): International Conference on Intelligent User Interfaces 2004. January 13-16, 2004, Funchal, Madeira, Portugal.

G

[Gae05] GAEDKE, M., NUSSBAUMER, M., *Web Composition with WCAG in mind*, Proceedings of the International Cross-Disciplinary Workshop on Web Accessibility 2005 W4A, at the World Wide Web Conference (WWW2005), Chiba, Japan, May 2005.

[Gal96] GALITZ,W. O., *The Essential Guide to User Interface Design an introduction to GUI design principles and techniques*, John Wiley & Sons Edition, 1996.

[Gra05] MICHOTTE, B., *GrafiXML*. Accessible à http://www.usixml.org

[Goo05] GOOGLE, consulté le 02-02-2005. Accessible à http://www.google.com.

H

[Her98] HERMSDORF, D., GAPPA, H., PIEPER, M., *WebAdapter: A Prototype of a WWWBrowser with New Special Needs Adaptations*. In: Proceedings of the ERCIM Workshop "User Interfaces for All : Towardsam Accessible Web", Langholmen, Stockholm 1998.

[Hou03] HOUCHARD, B., *Synthèse n° 79, « 2003, année européenne des handicapés : le test de l'accès à l'emploi »*, consulté le 18-08-2005. Accessible à http://www.robert-schuman.org/Synth79.htm

[HSS05] *United States Department of Health & Human Services*, consulté le 03-07-2005. Accessible à http://www.hhs.gov/

[Hya01] HYATT, D., GOODGER, B.,HICKSON, I., WATERSON, C., *XML User Interface Language (XUL) Specification 1.0. WorldWideWeb*, 2001. Accessible à http://www.mozilla.org/projects/xul/.

I

[Ibm92] IBM, *Object-Oriented Interface Design: IBM Common User Access guidelines*. IBM, 1992.

[Ibm00] IBM, *IBM Guidelines for Writing Accessible Applications Using 100% Pure Java™*, Août 2000, consulté le 12-05-2005. Accessible à http://www-306.ibm.com/able/guidelines/java/snsjavag.html

[Ibm05a] IBM, *History of IBM Accessibility*,2005, consulté le 04-08-2005. Accessible à http://www-306.ibm.com/able/access_ibm/history.html

[Ibm05b] IBM, *Developer guidelines* ,2005, consulté le 07-07-2005. Accessible à http://www-306.ibm.com/able/guidelines/index.html

[Ibm05c] IBM, *IBM Home Page Reader 3.04 : The voice of the World Wide Web*, consulté le 19-08-2005. Accessible à http://www-306.ibm.com/able/solution_offerings/hpr.html

[IEA05] IEA, *La définition de l'ergonomie par l'association internationale de l'ergonomie*, consulté le 15-08-2005. Accessible à http://www.ergonomie-self.org/ergo/defergo.html

[Ifi05] IFIP, *International Federation for Information Processing*, 2005. Accessible à http://www.ifip.or.at/

[INDA05] IRISH NATIONAL DISABILITY AUTHORITY , *Irish National Disability Authority IT Accessibility Guidelines Version 1.1*, consulté le 24-07-2005. Accessible à http://accessit.nda.ie/technologyindex_1.html

[Ivo01] IVORY, M. Y., HEARST, M. A. *The state of the art in automating usability evaluation of user interfaces*. ACM Computing Surveys, 33(4), pp. 470–516, December 2001.

J

[Jan95] JANSEN, R., *Webxref*, Octobre 1995, consulté le 21-08-2005. Accessible à www.w3.org/Tools/webxref.html

[Jac05] JACOBS, I., *About W3C : History*, mise à jour en juin 2005. Accessible à http://www.w3.org/Consortium/history

K

[Kas05] KASDAY, L., BOHMAN, P., ANDERSON, S., MATURI, N., VARANASI, B., *WAVE 3.1 – Web Accessibility Versatile Evaluator*, consulté le 25-01-2005. Accessible à http://www.wave.webaim.org/wave/

[Kin05] KINOA, *(anciennement visualfriendly)*, consulté le 19-08-2005. Accessible à http://www.visualfriendly.com/fr/

[Kot02] KOTHARI, R., BASAK, J., *Perceptually Motivated Measures for Capturing Proximity of Web Page Elements: Towards Automated Evaluation of Web Page Layouts*, IBM India Research Laboratory, India, 2002. Accessible à http://www.www2002.org/CDROM/alternate/688/

[Koy03] KOYANI, S. J., BALLEY, R. W., NALL, J. R., *Research – Based Web Design & Usability Guidelines*, Octobre 2003, consulté le 1-08-2005. Accessible à http://usability.gov/pdfs/guidelines.html

L

[Lep02] LEPORINI, B., PATERNO, F., *Criteria for Usability of Accessible Web Sites*, Department of Computer Science, University of Pisa, Italy, 2002.

[Lif05] UsableNet, *LIFT, accessibility evaluation tool*, *Lift for Macromedia® Dreamweaver®*, consulté le 14-08-2005. Accessible à http://www.usablenet.com/products_services/lift_dw/lift_dw.html

[Limb04] LIMBOURG, Q., VANDERDONCKT, J., *UsiXML: A User Interface Description Language Supporting Multiple Levels of Independence*, in Matera, M., Comai, S. (Eds.), «Engineering Advanced Web Applications», Rinton Press, 2004.

M

[Mah97] MAHAJAN, R., SHNEIDERMAN, B., *Sherlock*, 1997.

[Mar87] MARSHALL, C., NELSON, C., GARDINER, M.M., Design guidelines. In *Applying CognitivePsychology to User- Interface Design*, eds. M. M. Gardiner and B. Christie, Chichester: Wiley & Sons Ltd, New York, USA, 1987, pages 221-278.

[Mar05] MARGETTS, D., *UI Hall of Shame*, consulté le 15-08-2005. Accessible à http://www.userinterfacehallofshame.com/.

[May92] MAYHEW, D.J., P*rinciples and Guidelines in Software User Interface Design*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[Mic95] MICROSOFT, *The Windows Interface Guidelines for Software Design*. Microsoft Press, 1995.

[Mic05] MICROSOFT, *Microsoft Active Accessibility*, consulté le 02-08-2005. Accessible à http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msaa/msaastart_9w2t.asp

[Mor05] MORIN, Y., *Validateur HTML/XTML*, consulté le 10-08-2005. Accessible à http://yansanmo.no-ip.org:8080/ysm-validator/check.php

[Mur87] MURCH, G. M., *Color graphics - Blessing or Ballyhoo ? The Visual Channel*. In Baecker, R.M. and Buxton, W.A.S. eds. *Readings in Human-Computer Interaction - A Multidisciplinary Approach*. Morgan Kaufmann Publishers, 1987, pages 333–341.

N

[New00] NEWMAN, C., *Considering the Color-Blind*, août 2000. Accessible à http://webtechniques.com/archives/2000/08/newman/

[Ngo00] NGO, D. C. L., TEO, L. S., *A Mathematical Theory of Interface Aesthetics*,2000. Accessible à http://www.mi.sanu.ac.yu/vismath/ngo/

[Nie93] NIELSEN, J., *Usability Engineering*, San Diego, CA: Academic Press, 1993.

[Nog02] NOGIER, J-F, *De l'ergonomie du logiciel au design de site web*, Dunod, Paris, 2002.

[Nog05] NOGIER, J-F., *Usabilis, Evaluation ergonomique (usabilis)*, consulté le 15-08-2005. Accessible à http://www.usabilis.com/methode/evaluation_ergonomique.htm

[Nul05] Nullsoft, *Winamp.com – Skin*, consulté le 07-08-2005. Accessible à http://www.winamp.com/skins/

O

[Oca05] OCAL, K., *Etude et développement d'un interpréteur Usixml en Java*. Mémoire, 2005.

P

[Pate99] PATERNO, F., *Model Based Design and Evaluation of Interactive Applications*. Springer Verlag, Berlin, 1999.

[Pil03] PILGRIM, M., *He who casts the first stone,* Octobre 2003, consulté le 16-08-2005. Accessible à www.webstandards.org/buzz/archive/2003_10.html

[Pue02] PUERTA, A.,EISENSTEIN, J., *"XIML: A Common Representation for Interaction Data"*, Proceedings. of the 7th international conference on Intelligent user interfaces (San Francisco, 14-16 January), ACM Press, New York, USA, 2002.

[Per04] PERBEN, J., *Pour un Internet plus accessible aux personnes handicapées*, février 2004, consulté le 06-08-2005. Accessible à http://www.zdnet.fr/actualites/telecoms/0,39040748,39141986,00.htm

R

[Rag03] RAGGETT, D., *Clean up your Web pages with HTML TIDY*, 2003, consulté le 07-08-2005. Accessible à http://www.w3.org/People/Raggett/tidy/.

S

[Sch01] SCHWARZ, E., HENAULT, G., BURGER, D., *BrailleSurf 4*, 2001. Accessible à http://www.snv.jussieu.fr/inova/bs4/uk/

[Smi86] SMITH S.L., MOSIER J.N, *Guidelines for designing user interface software*, Report EDS-TR-86-278, The MITRE Corporation, Bedford, Massachusetts, 1986.

[Sno05] SNOOK, J., *Color Contrast Check*, Janvier 2005, consulté le 15-08-2005. Accessible à http://www.snook.ca/technical/colour_contrast/colour.html

T

[Tho05] THOA, E., *Le concept d'accessibilité*, Avril 2005, consulté le 04-06-2005. Accessible à http://www.ergologique.com/

U

[UKEGU02] U.K. E-Government Unit, Illustrated *Handbook for Web Management Teams*, 2002, consulté le 24-07-2005. Accessible à http://www.cabinetoffice.gov.uk/e-government/resources/handbook/

V

[Vand05] VANDERDONCKT, J, *LINF2356 - Interfaces homme-machine*, 2004-2005, à l'Université Catholique de Louvain.

[Vand04] VANDERDONCKT, J., BEIREKDAR, A., NOIRHOMME-FRAITURE M., *Automated Evaluation of Web Usability and Accessibility by Guideline Review*. International Conference on Web Engineering , Juillet 2004, Munich.

W

[W3C00a] W3C, *Core Techniques for Web Content Accessibility Guidelines 1.0*, Novembre 2000, consulté le 20-08-2005. Accessible à http://www.w3.org/TR/WCAG10-CORE-TECHS/

[W3C00b] W3C, *HTML Techniques for Web Content Accessibility Guidelines 1.0*, Novembre 2000, consulté le 20-08-2005. Accessible à http://www.w3.org/TR/WCAG10-HTML-TECHS/

[W3C00c] W3C, *CSS Techniques for Web Content Accessibility Guidelines 1.0*, Novembre 2000, consulté le 20-08-2005. Accessible à http://www.w3.org/TR/WCAG10-CSS-TECHS/

[W3C05] W3C, *The W3C Markup Validation Service*. Accessible à http://validator.w3.org/

[WAI98] WAI Accessibility Guidelines, *Web Accessibility Initiative*, World Wide Web Consortium, Genève, 1998. Accessible à http://www.w3.org/wai

[WAI04] WAI, *How People with Disabilities Use the Web*, Décembre 2004, consulté le 2-08-2005. Accessible à http://www.w3.org/WAI/EO/Drafts/PWD-Use-Web/Overview.html

[Wat04a] WATCHFIRE WebXACT, 2004, consulté le 05-08-2005. Accessible à http://www.webxact.com/

[Wat04b] WATCHFIRE WebXACT, 2004, *Welcome to WebXACT Help*, consulté le 12-08-2005. Accessible à http://webxact.watchfire.com/themes/standard-en-us/help/Server_default.html

[Wata04] WATANABE, T., *Web accessibility standards and policies in Japan*, Novembre 2004, consulté le 24-07-2005. Accessible à http://www.comm.twcu.ac.jp/~nabe/data/ITRC16.

[WCAG99a] WAI, *Web Content Accessibility Guidelines 1.0*, Mai 1999, consulté le 06-08-2005. Accessible à http://www.w3.org/TR/WAI-WEBCONTENT/

[WCAG99b] WAI, *Web Content Accessibility Guidelines 1.0 - 3. How the Guidelines are Organized*, Mai 1999. Accessible à http://www.w3.org/TR/WCAG10/#organization

[WCAG99c] WAI, *Web Content Accessibility Guidelines 1.0 - 4. Priorities*, Mai 1999. Accessible à http://www.w3.org/TR/WCAG10/#priorities

[WCAG05] WAI, *Web Content Accessibility Guidelines 2.0 Working Draft*, Juin 2005, consulté le 06-08-2005. Accessible à http://www.w3.org/TR/WCAG20/

X

[Xsl99] W3C, *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, Novembre 1999. Accessible à http://www.w3.org/TR/xslt.

[XPa99] W3C, *XML Path Language (XPath) Version 1.0* W3C Recommendation, Novembre 1999. Accessible à http://www.w3.org/TR/xpath

[XQu05] W3C, *XQuery 1.0: An XML Query Language W3C*, Avril 2005. Accessible à http://www.w3.org/TR/xquery

[XPo02] W3C, *XML Pointer Language (XPointer), W3C Working Draft*, August 2002. Accessible à http://www.w3.org/TR/xptr

Chapitre 9 Annexes

9.1 Web Content Accessibility Guidelines 1.0

1 Guideline : Provide equivalent alternatives to auditory and visual content.

Provide content that, when presented to the user, conveys essentially the same function or purpose as auditory or visual content.

Checkpoints:

1.1 Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content). This includes: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video.

1.2 Provide redundant text links for each active region of a server-side image map.

1.3 Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation.

1.4 For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation.

1.5 Until user agents render text equivalents for client-side image map links, provide redundant text links for each active region of a client-side image map.

2 Guideline : Don't rely on color alone.

Ensure that text and graphics are understandable when viewed without color.

Checkpoints:

2.1 Ensure that all information conveyed with color is also available without color, for example from context or markup.

2.2 Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen.

3 Guideline : Use markup and style sheets and do so properly.

Mark up documents with the proper structural elements. Control presentation with style sheets rather than with presentation elements and attributes.

Checkpoints:

3.1 When an appropriate markup language exists, use markup rather than images to convey information.
3.2 Create documents that validate to published formal grammars.
3.3 Use style sheets to control layout and presentation.
3.4 Use relative rather than absolute units in markup language attribute values and style sheet property values.
3.5 Use header elements to convey document structure and use them according to specification.
3.6 Mark up lists and list items properly.
3.7 Mark up quotations. Do not use quotation markup for formatting effects such as indentation.

4 Guideline 4. Clarify natural language usage

Use markup that facilitates pronunciation or interpretation of abbreviated or foreign text.

Checkpoints:

4.1 Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions).
4.2 Specify the expansion of each abbreviation or acronym in a document where it first occurs.
4.3 Identify the primary natural language of a document.

5 Guideline : Create tables that transform gracefully.

Ensure that tables have necessary markup to be transformed by accessible browsers and other user agents.

Checkpoints:

5.1 For data tables, identify row and column headers.
5.2 For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells.
5.3 Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent (which may be a linearized version).
5.4 If a table is used for layout, do not use any structural markup for the purpose of visual formatting
5.5 Provide summaries for tables.
5.6 Provide abbreviations for header labels.

6 Guideline : Ensure that pages featuring new technologies transform gracefully.

Ensure that pages are accessible even when newer technologies are not supported or are turned off.

Checkpoints:

6.1 Organize documents so they may be read without style sheets.

6.2 Ensure that equivalents for dynamic content are updated when the dynamic content changes.
6.3 Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page.
6.4 For scripts and applets, ensure that event handlers are input device-independent.
6.5 Ensure that dynamic content is accessible or provide an alternative presentation or page.

7 Guideline : Ensure user control of time-sensitive content changes.
Ensure that moving, blinking, scrolling, or auto-updating objects or pages may be paused or stopped.

Checkpoints:
7.1.1 Until user agents allow users to control flickering, avoid causing the screen to flicker.
7.1.2 Until user agents allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off).
7.1.3 Until user agents allow users to freeze moving content, avoid movement in pages.
7.1.4 Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages.
7.1.5 Until user agents provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects.

8 Guideline : Ensure direct accessibility of embedded user interfaces.
Ensure that the user interface follows principles of accessible design: device-independent access to functionality, keyboard operability, self-voicing, etc.

Checkpoint:
8.1 Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies.

9 Guideline : Design for device-independence.
Use features that enable activation of page elements via a variety of input devices.

Checkpoints:
9.1 Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape.
9.2 Ensure that any element that has its own interface can be operated in a device-independent manner.
9.3 For scripts, specify logical event handlers rather than device-dependent event handlers.
9.4 Create a logical tab order through links, form controls, and objects.
9.5 Provide keyboard shortcuts to important links (including those in client-side image maps), form controls, and groups of form controls.

10 Guideline : Use interim solutions.
Use interim accessibility solutions so that assistive technologies and older browsers will operate correctly.

Checkpoints:
10.1 Until user agents allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user.
10.2 Until user agents support explicit associations between labels and form controls, for all form controls with implicitly associated labels, ensure that the label is properly positioned.
10.3 Until user agents (including assistive technologies) render side-by-side text correctly, provide a linear text alternative (on the current page or some other) for all tables that lay out text in parallel, word-wrapped columns.
10.4 Until user agents handle empty controls correctly, include default, place-holding characters in edit boxes and text areas.
10.5 Until user agents (including assistive technologies) render adjacent links distinctly, include non-link, printable characters (surrounded by spaces) between adjacent links.

11 Guideline : Use W3C technologies and guidelines.
Use W3C technologies (according to specification) and follow accessibility guidelines. Where it is not possible to use a W3C technology, or doing so results in material that does not transform gracefully, provide an alternative version of the content that is accessible.

Checkpoints:
11.1 Use W3C technologies when they are available and appropriate for a task and use the latest versions when supported.
11.2 Avoid deprecated features of W3C technologies.
11.3 Provide information so that users may receive documents according to their preferences (e.g., language, content type, etc.)
11.4 If, after best efforts, you cannot create an accessible page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent information (or functionality), and is updated as often as the inaccessible (original) page.

12 Guideline : Provide context and orientation information.
Provide context and orientation information to help users understand complex pages or elements.

Checkpoints:
12.1 Title each frame to facilitate frame identification and navigation.
12.2 Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone.

12.3 Divide large blocks of information into more manageable groups where natural and appropriate.
12.4 Associate labels explicitly with their controls.

13 Guideline : Provide clear navigation mechanisms.
Provide clear and consistent navigation mechanisms -- orientation information, navigation bars, a site map, etc. -- to increase the likelihood that a person will find what they are looking for at a site.

Checkpoints:
13.1 Clearly identify the target of each link.
13.2 Provide metadata to add semantic information to pages and sites.
13.3 Provide information about the general layout of a site.
13.4 Use navigation mechanisms in a consistent manner.
13.5 Provide navigation bars to highlight and give access to the navigation mechanism.
13.6 Group related links, identify the group (for user agents), and, until user agents do so, provide a way to bypass the group.
13.7 If search functions are provided, enable different types of searches for different skill levels and preferences.
13.8 Place distinguishing information at the beginning of headings, paragraphs, lists, etc.
13.9 Provide information about document collections (i.e., documents comprising multiple pages.).
13.10 Provide a means to skip over multi-line ASCII art.

14 Guideline : Ensure that documents are clear and simple.
Ensure that documents are clear and simple so they may be more easily understood.

Checkpoints:
14.1 Use the clearest and simplest language appropriate for a site's content.
14.2 Supplement text with graphic or auditory presentations where they will facilitate comprehension of the page.
14.3 Create a style of presentation that is consistent across pages.

9.2 Web Content Accessibility Guidelines 2.0

1 Principle: Content must be perceivable.

1.1 Guideline: Provide text alternatives for all non-text content.

Level 1 Success Criteria

- For all non-text content that is used to convey information, text alternatives identify the non-text content and convey the same information. For multimedia, provide a text-alternative that identifies the multimedia.
- For functional non-text content, text alternatives serve the same purpose as the non-text content. If text alternatives can not serve the same purpose as the functional non-text content, text alternatives identify the purpose of the functional non-text content
- For non-text content that is intended to create a specific sensory experience, text alternatives at least identify the non-text content with a descriptive label.
- Non-text content that is not functional, is not used to convey information, and does not create a specific sensory experience is implemented such that it can be ignored by assistive technology.
- For live audio-only or live video-only content, text alternatives at least identify the purpose of the content with a descriptive label.

Level 2 Success Criteria

- No level 2 success criteria for this guideline.

Level 3 Success Criteria

- For prerecorded multimedia content, a combined transcript of captions and audio descriptions of video is available.

1.2 Guideline: Provide synchronized alternatives for multimedia.

Level 1 Success Criteria

- Captions are provided for prerecorded multimedia.

Level 2 Success Criteria

- Real-time captions are provided for live multimedia.

Level 3 Success Criteria

- Sign language interpretation is provided for multimedia
- Extended audio descriptions of video are provided for prerecorded multimedia.
- Audio descriptions of video are provided for live multimedia.

1.3 Guideline: Ensure that information, functionality, and structure can be separated from presentation.

Level 1 Success Criteria

- Structures within the content can be programmatically determined.

- When information is conveyed by color, the color can be programmatically determined or the information is also conveyed through another means that does not depend on the user's ability to differentiate colors.

Level 2 Success Criteria

- Information that is conveyed by variations in presentation of text is also conveyed in text or the variations in presentation of text can be programmatically determined.
- Any information that is conveyed by color is visually evident when color is not available.

Level 3 Success Criteria

- When content is arranged in a sequence that affects its meaning, that sequence can be determined programmatically.

1.4 Guideline: Make it easy to distinguish foreground information from background images or sounds.

Level 1 Success Criteria

- Any text that is presented over a background image, color, or text can be programmatically determined.

Level 2 Success Criteria

- Text and diagrams that are presented over a background image, color, or text have a contrast greater than X1 where the whiter element is at least Y1 as measured by _____.
- Text that is presented over a background pattern of lines which are within 500% +/- of the stem width of the characters or their serifs must have a contrast between the characters and the lines that is greater than X2, where the whiter element is at least Y2.
- A mechanism is available to turn off background audio that plays automatically.

Level 3 Success Criteria

- Text is not presented over any background (image, text, color or pattern), or if any background is present, the contrast between the text and the background is greater than X2.
- Audio content does not contain background sounds or the background sounds are at least 20 decibels lower than the foreground audio content, with the exception of occasional sound effects.

2 Principle: Interface elements in the content must be operable.

2.1 Guideline: Make all functionality operable via a keyboard interface.

Level 1 Success Criteria

- All of the functionality of the content, where the functionality or its outcome can be described in a sentence, is operable through a keyboard interface.

Level 2 Success Criteria

- No level 2 success criteria for this guideline.

Level 3 Success Criteria

- All functionality of the content is designed to be operated through a keyboard interface.

2.2 Guideline: Allow users to control time limits on their reading or interaction.

Level 1 Success Criteria for Guideline 2.2

- Content is designed so that time-outs are not an essential part of interaction, or at least one of the following is true for each time-out that is a function of the content:
  - the user is allowed to deactivate the time-out or;
  - the user is allowed to adjust the time-out over a wide range which is at least ten times the length of the default setting or;
  - the user is warned before time expires, allowed to extend the time-out with a simple action (for example, "hit any key") and given at least 20 seconds to respond or;
  - the time-out is an important part of a real-time event (for example, an auction), and no alternative to the time-out is possible or;
  - the time-out is part of an activity where timing is essential (for example, competitive gaming or time-based testing) and time limits can not be extended further without invalidating the activity.

Level 2 Success Criteria for Guideline 2.2

- Content does not blink for more than 3 seconds, or a method is available to stop any blinking content in the delivery unit.
- Moving or time-based content can be paused by the user.

Level 3 Success Criteria for Guideline 2.2

- Except for real-time events, timing is not an essential part of the event or activity presented by the content.
- Non-emergency interruptions, such as the availability of updated content, can be postponed or suppressed by the user.
- When an authenticated session has an inactivity timeout, the user can continue the activity without loss of data after re-authenticating

2.3 Guideline: Allow users to avoid content that could cause seizures due to photosensitivity.

Level 1 Success Criteria for Guideline 2.3

- Content that violates Computer adapted ITC Guidance for general flash or Computer adapted ITC Guidance for red flash is marked in a way that the user can avoid its appearance.

Level 2 Success Criteria for Guideline 2.3

- Content does not violate Computer adapted ITC Guidance for general flash or Computer adapted ITC Guidance for red flash.

Level 3 Success Criteria for Guideline 2.3

- Content does not violate Computer adapted spatial pattern thresholds or red flash.

2.4 Guideline: Provide mechanisms to help users find content, orient themselves within it, and navigate through it.

Level 1 Success Criteria for Guideline 2.4

- Navigational features can be programmatically identified.

Level 2 Success Criteria for Guideline 2.4

- More than one way is available to locate content within a set of delivery units.
- Blocks of content that are repeated on multiple perceivable units are implemented so that they can be bypassed.
- Delivery units have descriptive titles
- The destination of each programmatic reference to another delivery unit is identified through words or phrases that either occur in text or can be programmatically determined.

Level 3 Success Criteria for Guideline 2.4

- When a page or other delivery unit is navigated sequentially, elements receive focus in an order that follows relationships and sequences in the content.
- Information about the user's location within a set of delivery units is available.

2.5 Guideline: Help users avoid mistakes and make it easy to correct them.

Level 1 Success Criteria

- No level 1 success criteria for this guideline.

Level 2 Success Criteria

- If an input error is detected, the error is identified and provided to the user in text.
- If an input error is detected and suggestions for correction are known and can be provided without jeopardizing the security or purpose of the content, the error is identified and the suggestions are provided to the user.
- For forms that cause legal or financial transactions to occur, that modify or delete data in remote data storage systems, or that submit test responses, at least one of the following is true:
    - Actions are reversible.
    - Actions are checked for errors before going on to the next step in the process.
    - The user is able to review and confirm or correct information before submitting it.

Level 3 Success Criteria

- Additional context-relevant assistance is available for text input.

3 Principle: Content and controls must be understandable.

3.1 Guideline: Make text content readable and understandable.

Level 1 Success Criteria

- The primary natural language or languages of the delivery unit can be programmatically determined.

Level 2 Success Criteria

- The natural language of each foreign passage or phrase in the content can be programmatically determined.

Level 3 Success Criteria

- A mechanism is available for finding definitions for all words in text content.
- A mechanism is available for identifying specific definitions of words used in an unusual or restricted way, including idioms and jargon.
- A mechanism for finding the expanded form of acronyms and abbreviations is available.
- Section titles are descriptive.
- When text requires reading ability at or above the upper secondary education level, one or more of the following supplements is available:
    - A text summary that requires reading ability no higher than primary education level.
    - Graphical illustrations of concepts or processes that must be understood in order to use the content.
    - A spoken version of the text content.

3.2 Guideline: Make the placement and functionality of content predictable.

Level 1 Success Criteria

- No level 1 success criteria for this guideline.

Level 2 Success Criteria

- Components that are repeated on multiple delivery units within a set of delivery units occur in the same order each time they are repeated.
- When any component receives focus, it does not cause a change of context.
- Changing the setting of any input field does not automatically cause a change of context.
- Components that have the same functionality in multiple delivery units within a set of delivery units are labeled consistently.

Level 3 Success Criteria

- Graphical components that appear on multiple pages, including graphical links, are associated with the same text equivalents wherever they appear.
- Changes of context are initiated only by user action.

4 Principle: Content must be robust enough to work with current and future technologies.

4.1 Guideline: Use technologies according to specification.

- No level success criteria for this guideline.

4.2 Guideline : Ensure that user interfaces are accessible or provide an accessible alternative(s)

Level 1 Success Criteria

- If content does not meet all level 1 success criteria, then an alternate form is provided that does meet all level 1 success criteria.
- Content using baseline technologies or non-baseline technologies, must meet the following criteria:
    - Content that violates international health and safety standards for general flash or red flash is marked in a way that the user can avoid its appearance
    - If the user can enter the content using the keyboard, then the user can exit the content using the keyboard.
- The role, state, and value can be programmatically determined for every user interface component of the web content that accepts input from the user or changes dynamically in response to user input or external events.
- The label of each user interface control that accepts input from the user can be programmatically determined and is explicitly associated with the control.
- The states and values of content that can be changed via the user interface can also be changed programmatically.
- Changes to content, structure, selection, focus, attributes, values, state, and relationships within the content can be programmatically determined.

Level 2 Success Criteria

- Accessibility conventions of the markup or programming language (API's or specific markup) are used.

Level 3 Success Criteria

- A content implemented using technology outside of baseline follows all WCAG requirements supported by the technology.

9.3 AccessiWeb

1. Eléments graphiques.
1.1 : Chaque élément graphique possède-t-il une alternative textuelle ? Niveau : Bronze

1.2 : Pour chacune des images de la page ayant une alternative, les textes dans l'attribut ALT sont-ils appropriés par rapport au contexte dans lequel l'image se trouve ? Niveau : Bronze

1.3 : Les éléments graphiques destinés à la décoration sont-ils commentés par ALT="" ? Niveau : Bronze

1.4 : Pour chacune des images de la page, les textes dans l'attribut ALT font-ils moins de 60 caractères ? Niveau : Bronze

1.5 : Les commentaires associés à chacune des zones réactives d'une image map sont-ils pertinents ? Niveau : Bronze

1.6 : Les zones de chacune des images MAP sont-elles ordonnées de manière logique ? Niveau : Argent

1.7 : Pour chacune des images MAP, les zones de l'image MAP sont-elles définies juste après la déclaration de l'image MAP ? Niveau : Argent

1.8 : Pour chacune des images texte de la page, le contenu de son alternative est-il au moins équivalent au texte inscrit dans l'image ? Niveau : Bronze

1.9 : Il convient de remplacer un texte sous forme d'image par un texte mis en forme. Cette règle est-elle respectée ? Niveau : Argent

1.10 : Quand une image nécessite une description détaillée, un commentaire texte lui est-il associé ? Niveau : Bronze

1.11 : Si une description détaillée de l'image est présente, son contenu est-il pertinent ? Niveau : Bronze

1.12 : Pour chacune des images liens, le texte contenu dans l'attribut ALT donne-t-il la fonction du lien ? Niveau : Bronze

1.13 : Est-ce que la taille utilisée pour chacune des images est appropriée par rapport au contexte dans lequel elle se trouve ? Niveau : Or

2. Cadres.
2.1 : Y a-t-il un attribut NAME ? Niveau : Bronze

2.2 : Les noms donnés aux cadres sont-ils pertinents ? Niveau : Bronze

2.3 : Y a t-il une balise NOFRAME ? Niveau : Bronze

2.4 : Le contenu de la balise NOFRAME est-il pertinent ? Niveau : Bronze

2.5 : Y a-t-il un attribut TITLE ? Niveau : Bronze

2.6 : L'attribut TITLE est-il pertinent ? Niveau : Bronze

2.7 : L'attribut LONGDESC est-il présent pour expliquer l'interaction entre les cadres ? Niveau : Argent

2.8 : L'attribut LONGDESC est-il pertinent ? Niveau : Argent

2.9 : Y a-t-il un maximum de trois cadres dans la page ? Niveau : Bronze

2.10 : Lorsqu'il y a des cadres, le défilement ("scrolling" en anglais) est-il automatique ? Niveau : Bronze

3. Couleurs.
3.1 : L'information donnée par la couleur est-elle aussi lisible lorsque les couleurs sont désactivées ? Niveau : Bronze

3.2 : Les différences de contrastes entre les couleurs sont-elles suffisamment élevées ? Niveau : Bronze

4. Multimédia.
4.1 : Est-il possible de récupérer les informations fournies dans les supports multimédias d'une autre manière ? Niveau : Bronze

4.2 : Le contenu multimédia est-il synchronisé avec son alternative ? Niveau : Bronze

5. Tableaux.
5.1 : L'attribut SUMMARY est-il présent et pertinent ? Niveau : Bronze

5.2 : Dans un tableau de données, la balise CAPTION est-elle utilisée pour donner un titre au tableau ? Niveau : Bronze

5.3 : Dans les tableaux de données, y a t-il des en-têtes de colonnes appropriés ? Niveau : Bronze

5.4 : Dans un tableau de données, y a t-il un attribut HEADERS présent pour relier chacune des cellules du tableau ? Niveau : Bronze

5.5 : Dans un tableau de données, lorsqu'un titre de colonne dépasse 15 caractères, l'attribut ABBR est-il utilisé ? Niveau : Argent

5.6 : Dans un tableau de mise en forme, le contenu est-il correctement ordonné ? Niveau : Bronze

6. Liens.
6.1 : L'intitulé des liens fait-il moins de 80 caractères ? Niveau : Bronze

6.2 : Les liens sont-ils explicites ? Niveau : Bronze

6.3 : Si nécessaire, l'attribut TITLE est-il présent et fait-il moins de 80 caractères ? Niveau : Bronze

6.4 : L'attribut TITLE donne-t-il plus d'informations concernant le lien que l'intitulé du lien lui-même ? Niveau : Bronze

6.5 : Chaque intitulé de lien identique amène t-il vers la même destination ? Niveau : Bronze

6.6 : Dans l'arborescence du site, y a t-il un maximum de 9 catégories par niveau de navigation ? Niveau : Or

6.7 : Y a-t-il moins de 40 liens actifs dans la page, hors liens nécessaires à la navigation ? Niveau : Or

7. Scripts.
7.1 : Si un script nécessite une alternative pour être accessible, l'information donnée par cette alternative est-elle équivalente à l'information fournie par le script ? Niveau : Bronze

7.2 : Des actions peuvent-elles être accomplies même si le périphérique pour lequel elles sont prévues est désactivé ? Niveau : Bronze

8. Eléments obligatoires.
8.1 : La balise DOCTYPE e st-elle présente au début du code source de la page ? Niveau : Bronze

8.2 : L'attribut LANG est-il présent au début du code source de la page pour identifier clairement la langue utilisée ? Niveau : Bronze

8.3 : Des éléments de description de la page sont-ils présentes en début de code source ? Niveau : Argent

8.4 : Existe-t-il une balise TITLE dans l'en tête de la page ? Niveau : Bronze

8.5 : Le contenu de la balise TITLE est-il explicite ? Niveau : Bronze

8.6 : Le contenu de la balise TITLE est-il différent d'une page à l'autre ? Niveau : Bronze

8.7 : Les changements de langue dans une page sont-ils signalés ? Niveau : Bronze

9. Structuration de l'information.
9.1 : Est-ce que la structuration de l'information est cohérente par rapport au contexte général du site ? Niveau : Bronze

9.2 : La page web est-elle structurée de manière cohérente ? Niveau : Bronze

9.3 : Y a-t-il un plan du site ? Niveau : Argent

9.4 : Y a-t-il une page d'aide expliquant les principes de navigation à l'intérieur du site ? Niveau : Argent

9.5 : A partir de n'importe quelle page du site, la page d'aide est-elle atteignable de manière identique ? Niveau : Argent

9.6 : Y a-t-il un moteur de recherche interne au site ? Niveau : Argent

9.7 : A partir de n'importe quelle page du site, le moteur de recherche est-il atteignable de manière identique ? Niveau : Argent

9.8 : La page de résultats du moteur de recherche comporte-t-elle au moins les éléments suivants : nombre maximum de réponses par page, nombre total de réponses, éléments de navigation ? Niveau : Or

10. Présentation de l'information.
10.1 : Le contenu de la page est-il séparé de sa présentation ? Niveau : Bronze

10.2 : Avec les feuilles de style désactivées, l'information est-elle toujours présente ? Niveau : Bronze

10.3 : Avec les feuilles de style désactivées, l'ordre d'apparition de l'information est-il respecté par rapport à l'ordre d'apparition initialement défini ? Niveau : Bronze

10.4 : Des valeurs relatives sont-elles utilisées pour dimensionner les tableaux et définir la taille des polices de caractère ? Niveau : Argent

10.5 : Si des valeurs absolues sont utilisées, le sont-elles sans conséquence sur l'affichage de l'information ? Niveau : Argent

10.6 : Est-ce que les polices de caractères présentes sur la page appartiennent à la famille de polices de caractères sans sérif ? Niveau : Argent

11. Formulaires.
11.1 : La balise LABEL et les attributs correspondants (ID, FOR) sont-ils présents ? Niveau : Bronze

11.2 : Les textes associés aux champs de formulaires donnent-ils leur fonction exacte ? Niveau : Argent

11.3 : Est-ce que la disposition des champs de formulaire par rapport aux textes qui leur sont associés ne pose aucune ambiguïté ? Niveau : Argent

11.4 : La balise FIELDSET est-elle présente pour encadrer des blocs d'information de même nature ? Niveau : Argent

11.5 : La balise LEGEND est-elle présente pour donner un titre au bloc d'informations encadré par la balise FIELDSET ? Niveau : Argent

11.6 : Dans un formulaire, le commentaire du bouton SUBMIT est-il pertinent ? Niveau : Bronze

11.7 : Le contrôle de saisie des champs du formulaire est-il accessible ? Niveau : Bronze

11.8 : Les informations sont-elles organisées dans un ordre logique dans les listes de choix ? Niveau : Argent

12. Aide à la navigation.
12.1 : La navigation dans l'ensemble des pages du site est-elle cohérente ? Niveau : Argent

12.2 : Le menu principal de navigation interne dans le site est-il toujours présent à la même place dans les pages ? Niveau : Bronze

12.3 : Existent-ils des barres de navigation qui facilitent l'accès pour la navigation interne dans le site ? Niveau : Or

12.4 : Y a-t-il des liens facilitant la navigation dans la page ? Niveau : Argent

12.5 : Les liens importants du site comportent-ils des raccourcis claviers ? Niveau : Or

12.6 : Si des raccourcis clavier ont été définis dans le site, sont-ils actifs dans la page ? Niveau : Bronze

12.7 : Y a-t-il des caractères de séparation lorsqu'il y a un groupement de liens ? Niveau : Or

12.8 : La page fait-elle moins de 70 Ko ? Niveau : Or

13. Contenus accessibles

13.1 : L'utilisateur a t-il le contrôle du rafraîchissement ? Niveau : Bronze

13.2 : Si une redirection automatique est présente, s'effectue-t-elle sans l'intermédiaire d'un script ? Niveau : Bronze

13.3 : Le visiteur est-il averti lorsque de nouvelles fenêtres apparaissent ? Niveau : Bronze

13.4 : Y a-t-il une alternative équivalente au script qui déclenche l'ouverture de nouvelles fenêtres ? Niveau : Bronze

13.5 : Y a-t-il des informations supplémentaires disponibles décrivant la consultation des fichiers en téléchargement ? Niveau : Bronze

13.6 : Lorsqu'un fichier peut être téléchargé, y a-t-il des formats alternatifs équivalents ? Niveau : Or

13.7 : Est-ce que la présentation spécifique d'une information n'entrave pas l'accès à son contenu ? Niveau : Bronze

13.8 : La présentation de la page est-elle réalisée sans détourner certaines balises de leur fonction d'origine ? Niveau : Argent

13.9 : Les balises ACRONYM et ABBR figurent-elles au moins sur le premier des acronymes, abréviations de même nature se trouvant dans la page ? Niveau : Or

13.10 : L'attribut TITLE de la balise ACRONYM est-il correctement rempli ? Niveau : Or

13.11 : Lorsqu'un acronyme est présent et que la balise ACRONYM n'est pas remplie, chaque lettre est-elle séparée par un point ? Niveau : Or

13.12 : Les éléments de textes (titres, sous-titres, phrases,...) ne doivent pas être écrits en lettres majuscules : cette règle est-elle vérifiée ? Niveau : Or

13.13 : La page fait-elle au maximum 3 écrans en résolution 1024*768 si aucune navigation interne n'est prévue ? Niveau : Or

9.4 BlindSurfer

CONTENU

1. Alternative textuelle pour tout élément non textual

2. Tableaux de données facilement lisibles

3. Alternatives pour les éléments multimédia (vidéo et audio)

4. Contenu compréhensible pour une personne qui ne peut distinguer les couleurs

5. Alternative accessible pour des pages ou présentations complexes

6. Indication des changements de langue

NAVIGATION

7. Couplage de chaque hyperlien avec un texte significatif

8. Bonnes possibilités de navigation

9. Implémentation correcte des cadres (frames)

10. Définition correcte des images cliquables (image maps)

OBJETS PROGRAMMATIQUES

11. Alternatives pour les objets programmatiques (Flash, Scripts, Applets, etc)

MISE EN PAGE

12. Efficacité des réglages des browsers

13. Mise en page correcte

14. Usage correct de fenêtres nouvelles et de fenêtres "pop-up"

FORMULAIRES

15. Conception correcte des formulaires

GENERALITES

16. Compatibilité avec les browsers

9.5 IBM Software Accessibility Checklist - Version 3.5.1

1. **Keyboard access**
   - 1.1. Provide keyboard equivalents for all actions.
   - 1.2. Do not interfere with keyboard accessibility features built into the operating system.
2. **Object information**
   - 2.1. Provide a visual focus indicator that moves among interactive objects as the input focus changes. This focus indicator must be programmatically exposed to assistive technology.
   - 2.2. Provide semantic information about user interface objects. When an image represents a program element, the information conveyed by the image must also be available in text.
   - 2.3. Associate labels with controls, objects, icons and images. If an image is used to identify programmatic elements, the meaning of the image must be consistent throughout the application.
   - 2.4. When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements and functionality required for completion and submission of the form, including all directions and cues.
3. **Sounds and multimedia**
   - 3.1. Provide an option to display a visual cue for all audio alerts.
   - 3.2. Provide accessible alternatives to significant audio and video.
   - 3.3. Provide an option to adjust the volume.
4. **Display**
   - 4.1. Provide text through standard system function calls or through an API (application programming interface) which supports interaction with assistive technology.
   - 4.2. Use color as an enhancement, not as the only way to convey information or indicate an action.
   - 4.3. Support system settings for high contrast for all user interface controls and client area content.
   - 4.4. When color customization is supported, provide a variety of color selections capable of producing a range of contrast levels.
   - 4.5. Inherit system settings for font, size, and color for all user interface controls.
   - 4.6. Provide an option to display animation in a non-animated presentation mode.
5. **Timing**
   - 5.1. Provide an option to adjust the response times on timed instructions or allow the instructions to persist.
   - 5.2. Do not use flashing or blinking text, objects, or other elements having a flash or blink frequency greater than 2 Hz and lower than 55 Hz.
6. **Verify accessibility**
   - 6.1. Test for accessibility using available tools.

9.6 IBM Web accessibility checklist - version 3.5

Checkpoint

1. Images and animations. Use the alt="text" attribute to provide text equivalents for images. Use alt="" for images that do not convey important information or convey redundant information.

2. Image maps. Use client-side image maps and alternative text for image map hot spots. If a server-side map is needed, provide equivalent text links.

3. Graphs and charts. Summarize the content of each graph and chart, or use the longdesc attribute to link to the description or data.

4. Multimedia. Provide captions or transcripts of important audio content. Provide transcripts or audio descriptions of important video content.

5. Scripts. Ensure the functionality of scripts is keyboard accessible. If the content affected by scripting is not accessible, provide an alternative.

6. Applets, plug-ins, and non-HTML content. When an applet, plug-in or other application is required to be present, provide a link to one that is directly accessible, or provide alternate content for those which are not directly accessible.

7. Forms. Make forms accessible to assistive technology.

8. Skip to main content. Provide methods for skipping over navigation links to get to main content of page.

9. Frames. Provide a title for each FRAME element and frame page. Provide an accessible source for each frame.

10. Table headers. Use the TH element to mark up table heading cells. Use the headers attribute on cells of complex data tables.

11. Cascading style sheets. Web pages should be readable without requiring style sheets.

12. Color & contrast. Ensure that all information conveyed with color is also conveyed in the absence of color.

13. Blinking, moving or flickering content. Avoid causing content to blink, flicker, or move.

14. Timed responses. When a timed response is required, alert the user, and give sufficient time to indicate more time is required.

15. Text-only page. If accessibility cannot be accomplished in any other way, provide a text-only page with equivalent information or functionality. Update the content of the text-only page whenever the primary page changes.

16. Verify accessibility. Test the accessibility using available tools.

9.7 IBM Java accessibility checklist - version 3.1

1. **Keyboard Access.**
   - 1.1. Provide keyboard equivalents for all actions.
   - 1.2. Do not interfere with keyboard accessibility features built into the operating system.
2. **Object information.**
   - 2.1. Implement the Java Accessibility API by:
     - using the Java Foundation Classes (JFC) / Swing components and/or
     - following the guidelines for " Building Custom Components" when extending the Java Foundation Classes and when implementing the Java Accessibility API on custom components.
   - 2.2. Set the focus.
   - 2.3. Set the name on all components and include the description on icons and graphics. If an image is used to identify programmatic elements, the meaning of the image must be consistent throughout the application.
   - 2.4. Associate labels with controls, objects, and icons.
3. **Sound and multimedia.**
   - 3.1. Provide an option to display a visual cue for all audio alerts.
   - 3.2. Provide accessible alternatives to significant audio and video.
   - 3.3. Provide an option to adjust the volume.
4. **Display.**
   - 4.1. Use color as an enhancement, not as the only way to convey information or indicate an action.
   - 4.2. Support system settings for high contrast for all user interface controls and client area content.
   - 4.3. When color customization is supported, provide a variety of color selections capable of producing a range of contrast levels.
   - 4.4. Support system settings for size, font and color for all user interface controls.
   - 4.5. Provide an option to display animation in a non-animated presentation mode.
5. **Timing.**
   - 5.1. Provide an option to adjust timed responses or allow the instruction to persist.
   - 5.2. Avoid the use of blinking text, objects, or other elements.
6. **Documentation.**
   - 6.1. Provide documentation in an accessible format.
   - 6.2. Provide documentation on all accessibility features, including keyboard access, as part of the regular product documentation.
7. **Verify accessibility**
   - 7.1. Test for accessibility using available tools.

9.8 Section 508 Standards

Subpart B -- Technical Standards

9.8.1 § 1194.21 Software applications and operating systems.

(a) When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.

(b) Applications shall not disrupt or disable activated features of other products that are identified as accessibility features, where those features are developed and documented according to industry standards. Applications also shall not disrupt or disable activated features of any operating system that are identified as accessibility features where the application programming interface for those accessibility features has been documented by the manufacturer of the operating system and is available to the product developer.

(c) A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that assistive technology can track focus and focus changes.

(d) Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.

(e) When bitmap images are used to identify controls, status indicators, or other programmatic elements, the meaning assigned to those images shall be consistent throughout an application's performance.

(f) Textual information shall be provided through operating system functions for displaying text. The minimum information that shall be made available is text content, text input caret location, and text attributes.

(g) Applications shall not override user selected contrast and color selections and other individual display attributes.

(h) When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.

(i) Color coding shall not be used as the only means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.

(j) When a product permits a user to adjust color and contrast settings, a variety of color selections capable of producing a range of contrast levels shall be provided.
(k) Software shall not use flashing or blinking text, objects, or other elements having a flash or blink frequency greater than 2 Hz and lower than 55 Hz.

(l) When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

9.8.2 § 1194.22 Web-based intranet and internet information and applications.

(a) A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).

(b) Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.

(c) Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.

(d) Documents shall be organized so they are readable without requiring an associated style sheet.

(e) Redundant text links shall be provided for each active region of a server-side image map.

(f) Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

(g) Row and column headers shall be identified for data tables.

(h) Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.

(i) Frames shall be titled with text that facilitates frame identification and navigation.

(j) Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.

(k) A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.

(l) When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.

(m) When a web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with §1194.21(a) through (l).

(n) When electronic forms are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

(o) A method shall be provided that permits users to skip repetitive navigation links.

(p) When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.

9.9 Research – Based Web Design & Usability Guidelines

1 Design Process and Evaluation

1.1 Set and State Goals

Identify and clearly articulate the primary goals of the website before beginning the design process.

1.2 Use an Iterative Design Approach

Develop and test prototypes through an iterative design approach to create the most useful and usable website.

1.3 Evaluate Websites Before and After Making Changes

Conduct "before and after" studies when revising a website to determine changes in usability.

1.4 Provide Useful Content

Provide content that is engaging, relevant, and appropriate to the audience.

1.5 Understand and Meet Users' Expectations

Ensure that the website format meets user expectations, especially related to navigation, content, and organization.

1.6 Establish User Requirements

Use all available resources to better understand users' requirements.

1.7 Use Parallel Design

Have several developers independently propose designs and use the best elements from each design.

1.8 Consider Many User Interface Issues

Consider as many user interface issues as possible during the design process.

1.9 Focus on Performance Before Preference

If user performance is important, make decisions about content, format, interaction, and navigation before deciding on colors and decorative graphics.

1.10 Set Usability Goals

Set performance goals that include success rates and the time it takes users to find specific information, or preference goals that address satisfaction and acceptance by users.

1.11 Select the Right Number of Participants

Select the right number of participants when using different usability techniques. Using too few may reduce the usability of a website; using too many wastes valuable resources.

1.12 Be Easily Found on the Web

In order to have a high probability of being accessed, ensure that a website is in the "top thirty" references presented from a major search engine.

1.13 Recognize Tester Bias

Recognize that a strong individual and group tester bias seems to exist when evaluating the usability of websites.

1.14 Use Heuristics Cautiously

Use heuristic evaluations and expert reviews with caution.

1.15 Use Cognitive Walkthroughs Cautiously

Use cognitive walkthroughs with caution.

1.16 Apply Automatic Evaluation Methods

Use appropriate 'automatic evaluation' methods to conduct initial evaluations on websites.

2 Optimizing the User Experience

2.1 Display Information in a Directly Usable Format

Display data and information in a format that does not require conversion by the user.

2.2 Do Not Display Unsolicited Windows or Graphics

Do not have unsolicited windows or graphics "pop-up" to users.

2.3 Provide Assistance to Users

Provide assistance for users who need additional help with the website.

2.4 Provide Printing Options

Provide a link to a complete printable or downloadable document if there are Web pages, documents, resources, or files that users will want to print or save in one operation.

2.5 Standardize Task Sequences

Allow users to perform tasks in the same sequence and manner across similar conditions.

2.6 Minimize Page Download Time

Minimize the time required to download a website's pages.

2.7 Warn of 'Time Outs'

Let users know if a page is programmed to 'time out,' and warn users before time expires so they can request additional time.

2.8 Reduce the User's Workload

Allocate functions to take advantage of the inherent respective strengths of computers and users.

2.9 Use Users' Terminology in Help Documentation

When giving guidance about using a website, use the users' terminology to describe elements and features.

2.10 Provide Feedback When Users Must Wait

Provide users with appropriate feedback while they are waiting.

2.11 Inform Users of Long Download Times

Indicate to users the time required to download an image or document at a given connection speed.

2.12 Do Not Require Users to Multitask While Reading

If reading speed is important, do not require users to perform other tasks while reading from the monitor.

2.13 Design For Working Memory Limitations

Do not require users to remember information from place to place on a website.

2.14 Develop Pages that Will Print Properly

If users are likely to print one or more pages, develop pages with widths that print properly.

3 Accessibility

3.1 Comply with Section 508

If a website is being designed for the United States government, ensure that it meets the requirements of Section 508 of the Rehabilitation Act. Ideally, all websites should strive to be accessible and compliant with Section 508.

3.2 Design Forms for Users Using Assistive Technologies

Ensure that users using assistive technology can complete and submit online forms.

3.3 Provide Text Equivalents for Non-Text Elements

Provide a text equivalent for every non-text element that conveys information.

3.4 Do Not Use Color Alone to Convey Information

Ensure that all information conveyed with color is also available without color.

3.5 Provide Equivalent Pages

Provide text-only pages with equivalent information and functionality if compliance with accessibility provisions cannot be accomplished in any other way.

3.6 Ensure that Scripts Allow Accessibility

When designing for accessibility, ensure that the information provided on pages that utilize scripting languages to display content or to create interface elements can be read by assistive technology

3.7 Provide Client-Side Image Maps

To improve accessibility, provide client-side image maps instead of server-side image maps.

3.8 Enable Users to Skip Repetitive Navigation Links

To aid those using assistive technologies, provide a means for users to skip repetitive navigation links.

3.9 Provide Frame Titles

To ensure accessibility, provide frame titles that facilitate frame identification and navigation.

3.10 Test Plug-ins and Applets for Accessibility

To ensure accessibility, test any applets, plug-ins or other applications required to interpret page content to ensure that they can be used by assistive technologies.

3.11 Synchronize Multimedia Elements

To ensure accessibility, provide equivalent alternatives for multimedia elements that are synchronized.

3.12 Do Not Require Style Sheets

Organize documents so they are readable without requiring an associated style sheet.

3.13 Avoid Screen Flicker

Design Web pages that do not cause the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.

4 Hardware and Software

4.1 Design for Common Browsers

Design, develop and test for the most common browsers.

4.2 Account for Browser Differences

Do not assume that all users will have the same browser features, and will have set the same defaults.

4.3 Design for Popular Operating Systems

Design the website so it will work well with the most popular operating systems.

4.4 Design for User's Typical Connection Speed

Design for the connection speed of most users.

4.5 Design for Commonly Used Screen Resolutions

Design for monitors with the screen resolution set at 800x600 pixels.

5 The Home Page

5.1 Create a Positive First Impression of Your Site

Treat your homepage as the key to conveying the quality of your site.

5.2 Ensure the Homepage Looks like a Homepage

Ensure that the homepage has the necessary characteristics to be easily perceived as a homepage.

5.3 Show All Major Options on the Homepage

Present all major options on the homepage.

5.4 Enable Access to the Homepage

Enable users to access the homepage from any other page on the website.

5.5 Attend to Homepage Panel Width

Ensure that homepage panels are of a width that will cause them to be recognized as panels.

5.6 Announce Changes to a Website

Announce major changes to a website on the homepage—do not surprise users.

5.7 Communicate the Website's Purpose

Communicate the purpose of the website on the homepage.

5.8 Limit Prose Text on the Homepage

Limit the amount of prose text on the homepage.

5.9 Limit Homepage Length

Limit the homepage to one screenful of information if at all possible.

6 Page Layout

6.1 Set Appropriate Page Lengths

Make page-length decisions that support the primary use of the Web page.

6.2 Use Frames When Functions Must Remain Accessible

Establish a high-to-low level of importance for information and infuse this approach throughout each page on the website.

6.3 Establish Level of Importance

Establish a high-to-low level of importance for information and infuse this approach throughout each page on the website.

6.4 Place Important Items at Top Center

Put the most important items at the top center of the Web page to facilitate users' finding the information.

6.5 Place Important Items Consistently

Put important, clickable items in the same locations, and closer to the top of the page, where their location can be better estimated.

6.6 Structure for Easy Comparison

Structure pages so that items can be easily compared when users must analyze those items to discern similarities, differences, trends, and relationships.

6.7 Use Moderate White Space

Limit the amount of white space (areas without text, graphics, etc.) on pages that are used for scanning and searching.

6.8 Align Items on a Page

Visually align page elements, either vertically or horizontally.

6.9 Choose Appropriate Line Lengths

If reading speed is most important, use longer line lengths (75-100 characters per line). If acceptance of the website is most important, use shorter line lengths (fifty characters per line).

6.10 Avoid Scroll Stoppers

Ensure that the location of headings and other page elements does not create the illusion that users have reached the top or bottom of a page when they have not.

7 Navigation

7.1 Provide Feedback on Users' Location

Provide feedback to let users know where they are in the website.

7.2 Use a Clickable 'List of Contents' on Long Pages

On long pages, provide a 'list of contents' with links that take users to the corresponding content farther down the page.

7.3 Do Not Create Pages with No Navigational Options

Do not create or direct users into pages that have no navigational options.

7.4 Differentiate and Group Navigation Elements

Clearly differentiate navigation elements from one another, but group and place them in a consistent and easy to find place on each page.

7.5 Use Descriptive Tab Labels

Ensure that tab labels are clearly descriptive of their function or destination.

7.6 Present Tabs Effectively

Ensure that navigation tabs are located at the top of the page, and look like clickable versions of real-world tabs.

7.7 Use Site Maps

Use site maps for websites that have many pages.

7.8 Use Appropriate Menu Types

Use 'sequential' menus for simple forward-moving tasks, and use 'simultaneous' menus for tasks that would otherwise require numerous uses of the Back button.

7.9 Keep Navigation-only Pages Short

Do not require users to scroll purely navigational pages.

7.10 Use 'Glosses' to Assist Navigation

Provide 'glosses' to help users to select correct links.

8 Scrolling and Paging

8.1 Eliminate Horizontal Scrolling

Use an appropriate page layout to eliminate the need for users to scroll horizontally.

8.2 Use Scrolling Pages For Reading Comprehension

Use longer, scrolling pages when users are reading for comprehension.

8.3 Use Paging Rather Than Scrolling

If users' system response times are reasonably fast, use paging rather than scrolling.

8.4 Scroll Fewer Screenfuls

If users are looking for specific information, break up the information into smaller portions (shorter pages).

8.5 Facilitate Rapid Scrolling

Facilitate fast scrolling by highlighting major items.

9 Headings, Titles, and Labels

9.1 Use Clear Category Labels

Ensure that category labels, including links, clearly reflect the information and items contained within the category.

9.2 Use Unique and Descriptive Headings

Use headings that are unique from one another and conceptually related to the content they describe.

9.3 Use Descriptive Row and Column Headings

Ensure that data tables have clear, concise and accurate row and column headings.

9.4 Use Descriptive Headings Liberally

Use descriptive headings liberally throughout a website.

9.5 Provide Descriptive Page Titles

Put a descriptive, unique, concise, and meaningfully different title on each Web page.

9.6 Highlight Critical Data

Visually distinguish (i.e., highlight) important page items that require user attention, particularly when those items are displayed infrequently.

9.7 Provide Users with Good Ways to Reduce Options

Provide users with good ways to reduce their available options as efficiently as possible.

9.8 Use Headings in the Appropriate HTML Order

Use headings in the appropriate HTML order.

10 Links

10.1 Provide Consistent Clickability Cues

Provide sufficient cues to clearly indicate to users that an item is clickable.

10.2 Avoid Misleading Cues to Click

Ensure that items that are not clickable do not have characteristics that suggest that they are clickable.

10.3 Use Text for Links

Use text links rather than image links.

10.4 Use Meaningful Link Labels

Use link labels and concepts that are meaningful, understandable, and easily differentiated by users rather than designers.

10.5 Match Link Names with Their Destination Pages

Make the link text consistent with the title or headings on the destination (i.e., target) page.

10.6 Ensure that Embedded Links are Descriptive

When using embedded links, the link text should accurately describe the link's destination.

10.7 Repeat Important Links

Ensure that important content can be accessed from more than one link.

10.8 Designate Used Links

Use color changes to indicate to users when a link has been visited.

10.9 Link to Related Content

Provide links to other pages in the website with related content.

10.10 Link to Supportive Information

Provide links to supportive information.

10.11 Use Appropriate Text Link Lengths

Make text links long enough to be understood, but short enough to minimize wrapping.

10.12 Indicate Internal vs. External Links

Indicate to users when a link will move them to a different location on the same page or to a new page on a different website.

10.13 Use 'Pointing-and-Clicking'

'Pointing-and-clicking', rather than 'mousing-over', is preferred when selecting menu items from a cascading menu structure.

10.14 Clarify Clickable Regions of Images

If any part of an image is clickable, ensure that the entire image is clickable or that the clickable sections are obvious.

11 Text Appearance

11.1 Use Black Text on Plain, High-Contrast Backgrounds

When users are expected to rapidly read and understand prose text, use black text on a plain, high-contrast, non-patterned background.

11.2 Ensure Visual Consistency

Ensure visual consistency of website elements within and between Web pages.

11.3 Format Common Items Consistently

Ensure that the format of common items is consistent from one page to another.

11.4 Use at Least 12-Point Font

Use at least a 12-point font (e.g., typeface) on all Web pages.

11.5 Use Familiar Fonts

Use a familiar font to achieve the best possible reading speed.

11.6 Emphasize Importance

Change the font characteristics to emphasize the importance of a word or short phrase.

11.7 Use Attention-Attracting Features when Appropriate

Draw attention to specific parts of a Web page with the appropriate (but limited) use of moving or animated objects, size differential between items, images, brightly-colored items, and varying font characteristics.

12 Lists

12.1 Order Elements to Maximize User Performance

Arrange lists and tasks in an order that best facilitates efficient and successful user performance.

12.2 Display Related Items in Lists

Display a series of related items in a vertical list rather than as continuous text.

12.3 Introduce Each List

Provide an introductory heading (i.e., word or phrase) at the top of each list.

12.4 Format Lists to Ease Scanning

Make lists easy to scan and understand.

12.5 Start Numbered Items at One

When items are numbered, start the numbering sequence at "one" rather than "zero."

12.6 Place Important Items at Top of the List

Place a list's most important items at the top.

12.7 Capitalize First Letter of First Word in Lists

Capitalize the first letter of only the first word of a list item, a list box item, check box labels, and radio button labels.

12.8 Use Appropriate List Style

Use bullet lists to present items of equal status or value, and numbered lists if a particular order to the items is warranted.

13 Screen-based Controls (Widgets)

13.1 Distinguish Required and Optional Data Entry Fields

Distinguish clearly and consistently between required and optional data entry fields.

13.2 Detect Errors Automatically

Use the computer to detect errors made by users.

13.3 Minimize User Data Entry

Do not require users to enter the same information more than once.

13.4 Label Data Entry Fields Clearly

Display an associated label for each data entry field to help users understand what entries are desired.

13.5 Put Labels Close to Data Entry Fields

Ensure that labels are close enough to their associated data entry fields so that users will recognize the label as describing the data entry field.

13.6 Label Pushbuttons Clearly

Ensure that a pushbutton's label clearly indicates its action.

13.7 Label Data Entry Fields Consistently

Ensure that data entry labels are worded consistently, so that the same data item is given the same label if it appears on different pages.

13.8 Allow Users to See Their Entered Data

Create data entry fields that are large enough to show all of the entered data without scrolling.

13.9 Display Default Values

Display default values whenever a likely default choice can be defined.

13.10 Use a Minimum of Two Radio Buttons

Never use one radio button alone.

13.11 Use Radio Buttons for Mutually Exclusive Selections

Provide radio buttons when users need to choose one response from a list of mutually exclusive options.

13.12 Use Check Boxes to Enable Multiple Selections

Use a check box control to allow users to select one or more items from a list of possible choices.

13.13 Use Familiar Widgets

Use widgets that are familiar to your users and employ them in their commonly used manner.

13.14 Use a Single Data Entry Method

Design data entry transactions so that users can stay with one entry method as long as possible.

13.15 Partition Long Data Items

Partition long data items into shorter sections for both data entry and data display.

13.16 Do Not Make User-Entered Codes Case Sensitive

Treat upper- and lowercase letters as equivalent when users are entering codes.

13.17 Place Cursor in First Data Entry Field

Place (automatically) a blinking cursor at the beginning of the first data entry field when a data entry form is displayed on a page.

13.18 Provide Auto-tabbing Functionality

Provide auto-tabbing functionality for frequent users with advanced Web interaction skills.

13.19 Label Units of Measurement

When using data entry fields, specify the desired measurement units with the field labels rather than requiring users to enter them.

13.20 Ensure that Double-Clicking Will Not Cause Problems

Ensure that double-clicking on a link will not cause undesirable or confusing results.

13.21 Do Not Limit Viewable List Box Options

When using open lists, show as many options as possible.

13.22 Use Open Lists to Select One from Many

Use open lists rather than drop-down (pull-down) lists to select one from many.

13.23 Prioritize Pushbuttons

Use location and highlighting to prioritize pushbuttons.

13.24 Minimize Use of the Shift Key

Design data entry transactions to minimize use of the Shift key.

13.25 Use Data Entry Fields to Speed Performance

Require users to enter information using data entry fields (instead of selecting from list boxes) if you are designing to speed human performance.

14 Graphics, Images, and Multimedia

14.1 Use Video, Animation, and Audio Meaningfully

Use video, animation, and audio only when they help to convey, or are supportive of, the website's message or other content.

14.2 Include Logos

Place your organization's logo in a consistent place on every page.

14.3 Limit Large Images Above the Fold

Do not fill the entire first screenful with one image if there are screenful of text information below the fold.

14.4 Limit the Use of Images

Use images only when they are critical to the success of a website.

14.5 Label Clickable Images

Ensure that all clickable images are either labeled or readily understood by typical users.

14.5.1 Ensure that Images Do Not Slow Downloads

Take steps to ensure that images on the website do not slow page download times unnecessarily.

14.5.2 Use Thumbnail Images to Preview Larger Images

When viewing full-size images is not critical, first provide a thumbnail of the image.

14.5.3 Graphics Should Not Look Like Banner Ads

Do not make important images look like banner advertisements or gratuitous decorations.

14.5.4 Use Simple Background Images

Use background images sparingly and make sure they are simple, especially if they are used behind text.

14.6 Include Actual Data with Data Graphics

Include actual data values with graphical displays of data when precise reading of the data is required.

14.7 Display Monitoring Information Graphically

Use a graphic format to display data when users must monitor changing data.

14.8 Introduce Animation

Provide an introductory explanation for animation prior to it being viewed.

14.9 Ensure Website Images Convey Intended Messages

Ensure that website images convey the intended message to users, not just to designers.

14.10 Use Images to Facilitate Learning

To facilitate learning, use images rather than text whenever possible.

14.11 Emulate Real-World Objects

Use images that look like real-world items when appropriate.

15 Writing Web Content

15.1 Define Acronyms and Abbreviations

Do not use unfamiliar or undefined acronyms or abbreviations on websites.

15.2 Use Abbreviations Sparingly

Show complete words rather than abbreviations whenever possible.

15.3 Use Familiar Words

Use words that are frequently seen and heard.

15.4 Use Mixed Case with Prose

Display continuous (prose) text using mixed upper- and lowercase letters.

15.5 Avoid Jargon

Do not use words that typical users may not understand.

15.6 Make First Sentences Descriptive

Include the primary theme of a paragraph, and the scope of what it covers, in the first sentence of each paragraph.

15.7 Use Active Voice

Compose sentences in active rather than passive voice.

15.8 Write Instructions in the Affirmative

As a general rule, write instructions in affirmative statements rather than negative statements.

15.9 Limit the Number of Words and Sentences

To optimize reading comprehension, minimize the number of words in sentences, and the number of sentences in paragraphs.

15.10 Limit Prose Text on Navigation Pages

Do not put a lot of prose text on navigation pages.

15.11 Make Action Sequences Clear

When describing an action or task that has a natural order or sequence (assembly instructions, troubleshooting, etc.), structure the content so that the sequence is obvious and consistent.

16 Content Organization

16.1.1.1 Organize Information Clearly

Organize information at each level of the website so that it shows a clear and logical structure to typical users.

16.1.1.2 Put Critical Information Near the Top of the Website

Put critical information high in the hierarchy of a website

16.1.1.3 Facilitate Scanning

Structure each content page to facilitate scanning: use clear, well-located headings; short phrases and sentences; and small readable paragraphs.

16.1.1.4 Group Related Elements

Group all related information and functions in order to decrease time spent searching or scanning.

16.1.1.5 Display Only Necessary Information

Limit page information only to that which is needed by users while on that page.

16.1.1.6 Ensure that Necessary Information is Displayed

Ensure that all needed information is available and displayed on the page where and when it is needed.

16.1.1.7 Format Information for Multiple Audiences

Provide information in multiple formats if the website has distinct audiences who will be interested in the same information.

16.1.2 Design Quantitative Content for Quick Understanding

Design quantitative information to reduce the time required to understand it.

16.1.3 Use Color for Grouping

Use color to help users understand what does and does not go together.

17 Search

17.1 Provide a Search Option on Each Page

Provide a search option on each page of a content-rich website.

17.2 Ensure Usable Search Results

Ensure that the results of user searches provide the precise information being sought, and in a format that matches users' expectations.

17.3 Allow Simple Searches

Structure the search engine to accommodate users who enter one or two keywords.

17.4 Make Upper- and Lowercase Search Terms Equivalent

Treat user-entered upper- and lowercase letters as equivalent when entered as search terms.

17.5 Design Search Engines to Search the Entire Site

Design search engines to search the entire site, or clearly communicate which part of the site will be searched.

17.6 Design Search Around Users' Terms

Construct a website's search engine to respond to users' terminology.

17.7 Notify Users When Multiple Search Options Exist

If more than one type of search option is provided, ensure that users are aware of all the different types of search options and how each is best used.

17.8 Provide Search Templates

Provide templates to facilitate the use of search engines.