HAI919I
Ingénierie Dirigée par les modèles

Rendu TP2

Auteur :

Canta Thomas
Desgenetez Charles
Fontaine Quentin
Reiter Maxime

Master 2 - Génie Logiciel
Faculté des sciences de Montpellier
Année universitaire 2021/2022

Travaux pratiques sur les options d'accessibilité visuelle de systèmes d'exploitation

✎ **Feature TextSize**

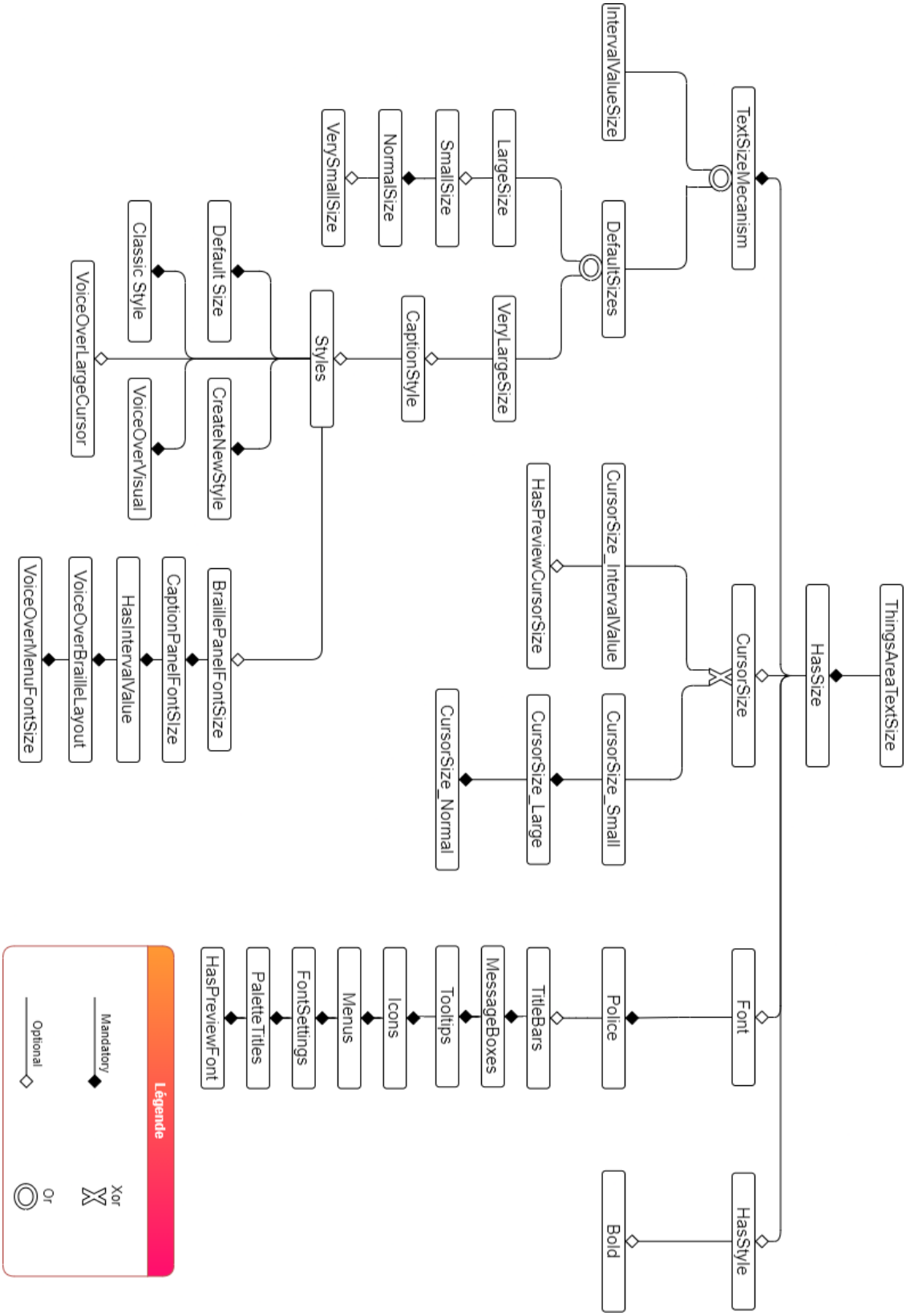

Fig. 1 – TextSize Feature Model

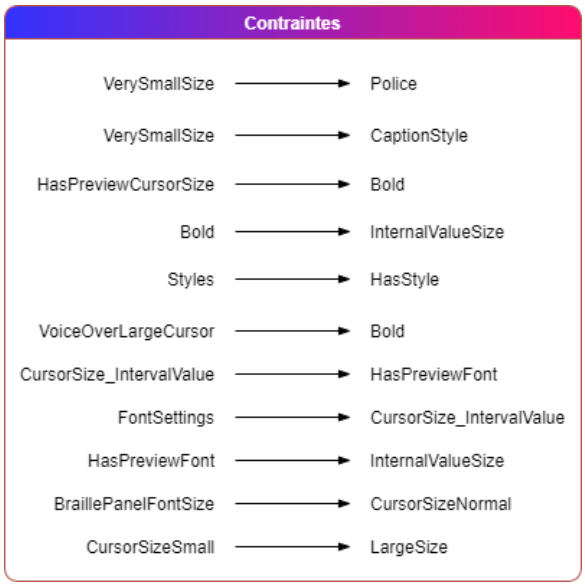

Fig. 2 – TextSize Feature Model Constraints

Avant d'initier un graphe nous avons dans un premier temps analysé et mis en œuvre des hypothèses pour obtenir une meilleure compréhension de nos données, notamment pour mettre en relation l'ontologie et le treillis. Notre première étape fût d'empaqueter et séparer les grands groupes visibles dans l'ontologie comme Size ou encore CursorSize pour obtenir une première approche de raisonnement. L'ontologie nous auras été utile pour le groupement des features, en revanche, le treillis nous as été plus bénéfique pour mettre en relation les différentes features.

Une fois cette étape de compréhension et de raisonnement terminée nous pouvions commencer la mise en œuvre d'un esquisse de notre feature model. Pour commencer, nous nous sommes, d'abord, concentré sur les features liées à la taille du texte *(LargeSize, VerySmallSize, IntervalValueSize...)*. Le treillis nous montre qu'un système d'exploitation (OS) comporte obligatoirement un mécanisme de taille de texte. Nous avons le choix entre des tailles prédéfinies, comme Large et Small que l'on décidera de regrouper dans une nouvelle feature *DefaultSize* (simplifiant la compréhension), ou bien (OR) avec un système d'intervalle de taille. Dans les tailles prédéfinies, le treillis nous montre que nous avons encore un choix, soit LargeSize, soit VeryLargeSize, ou les deux (OR). Mais encore, si l'OS implémente LargeSize, il peut implémenter SmallSize, si c'est le cas alors il a obligatoirement NormalSize. Cependant, si l'OS implémente LargeSize ou bien VeryLargeSize, il peut aussi avoir VerySmallSize, mais si c'est le cas alors l'OS doit avoir toutes les tailles prédéfinies (DefaultSize) comme le montre notre Feature Model et notre contrainte (VerySmallSize $\rightarrow$ VeryLargeSize).

Ce même type de raisonnement a été appliqué pour la feature CursorSize en regroupant au mieux les features qui lui son propre, car très éloignées sur le treillis. Nous avons mis un XOR principale sous notre CursorSize car soit l'on trouve CursorSize_IntervalValue (avec peut-être HasPreviewCursorSize) ou sinon il y'a les différentes tailles de curseurs déjà définis (CursorSize Small et cetera...)

Le reste du feature model sont les liaisons visibles dans le treillis. Lorsqu'une feature est reliée à une autre feature d'un concept différent alors nous les avons liées par un lien *Optional* (Optionnel). Lorsque que plusieurs features étaient dans le même concept, alors elles sont liées par un lien *Mandatory* (Obligatoire).

Les liaisons entre features qui ne sont pas représentées dans notre feature model apparaîtront donc sous forme de contraintes.