Description d'architectures en XML en vue de l'utilisation dans un système d'aide au déplacement des aveugles

Description d'architectures en XML

Christophe Jacquet[1] René Farcy[2] Yacine Bellik[1]

[1] LIMSI-CNRS, BP 133
91403 Orsay CEDEX
France
Nom.Prenom@limsi.fr
http://www.limsi.fr/

[2] Laboratoire Aimé Cotton
91405 Orsay CEDEX
France
Rene.Farcy@lac.u-psud.fr
http://www.lac.u-psud.fr/

Christophe Jacquet est stagiaire au LIMSI-CNRS, dans le cadre du DEA "Information, Interaction, Intelligence" de l'Université d'Orsay Paris XI et de la section "Systèmes Informatiques" de Supélec. Il travaille sur la modélisation d'environnements pour la conception d'un système d'aide au déplacement des aveugles.

René Farcy est Maître de conférences à l'Université d'Orsay Paris XI. Il enseigne l'optronique en formation d'ingénieur. Son thème de recherche porte sur l'instrumentation biomédicale (traitements lasers, orthèses visuelles pour les non-voyants).

Yacine Bellik est Maître de conférences en Informatique à l'Université d'Orsay Paris XI et prépare actuellement son habilitation à diriger des recherches. Il mène ses activités de recherche au sein du laboratoire LIMSI-CNRS, où il est responsable du thème "Interaction Multimodale". Ses travaux de recherche portent sur l'étude de la multimodalité en entrée et en sortie et sur ses applications dans le domaine du handicap visuel.

Résumé

Notre but premier est de concevoir un appareil d'aide au déplacement des personnes non voyantes, qui serait capable d'indiquer aux utilisateurs où ils se trouvent. Pour ce faire, nous devons concevoir une méthode pour décrire l'architecture des endroits parcourus. Ainsi, nous avons développé un modèle d'architecture, composé d'objets physiques ("lexicaux"), reliés pour former une structure logique ("syntaxique"), dont les éléments peuvent être agrégés. Nous dresserons d'abord une taxonomie des concepts mis en jeu, avant de décrire une méthode de représentation basée sur XML, avec des schémas XML. Enfin, nous montrerons comment notre représentation en XML peut être utilisée pour résoudre des problèmes comme la localisation d'un point.

Mots-clefs : XML, architecture, modèle, sémantique, schéma

Introduction

Dans cet article, nous supposerons que nous disposons d'un appareil capable de connaître à tout instant ses propres coordonnées géographiques, tel que le Télétact [1] [2], auquel on adjoindrait une centrale inertielle. Cet appareil est équipé d'un télémètre laser qui détecte la distance des obstacles. Nous voulons qu'il soit capable de savoir où il se trouve, *dans un environnement architectural connu*. De plus, quand l'utilisateur désigne un point grâce au pointeur laser, nous devons pouvoir récupérer des informations sémantiques pertinentes associées à ce point. C'est pourquoi nous avons besoin d'un modèle adéquat pour décrire les architectures.

Plusieurs standards industriels de description d'environnements 3D existent : parmi eux, VRML et X3D sont les plus connus. Cependant, ces formats basent les descriptions sur des formes géométriques simples. Au contraire, pour notre problème, nous avons besoin de traiter des objets de plus haut niveau, comme des étages, des murs, des pièces, etc. et pas seulement des formes géométriques. C'est pourquoi nous introduisons l'approche décrite ci-dessous.

1 Modélisation d'architecture

1.1 Aperçu du modèle

Nous décrivons un modèle en trois couches pour décrire les architectures :

Première couche : niveau lexical. Nous appelons "éléments lexicaux" les éléments architecturaux simples – c'est-à-dire élémentaires. En pratique, ce sont des murs, des portes, des volées de marche, etc.

Deuxième couche : niveau syntaxique. Les "éléments syntaxiques" sont complexes – c'est-à-dire composés. Ils sont constitués de l'arrangement de plusieurs "éléments lexicaux". Par exemple, une pièce est définie par ses murs.

Troisième couche : niveau d'agrégation. Les "éléments syntaxiques" sont alors rassemblés en "éléments d'agrégation". Ainsi, plusieurs bureaux peuvent être assemblés en un agrégat appelé, par exemple "service des ventes".

On peut faire un parallèle entre notre terminologie et la structure des textes en langage naturel. Au niveau le plus bas, les textes sont simplements écrits avec des mots : c'est le niveau lexical. Ces mots sont alors assemblés en phrases par rapport à une syntaxe définie. À leur tour, les phrases sont agrégées en diverses unités de sens telles que des paragraphes, des listes à puces, etc.

1.2 Hiérarchie de concepts

Nous utilisons une modélisation à objets pour décrire les concepts (ou *classes* de notre modèle. Pour des raisons de généricité, toutes les classes dérivent d'un ancêtre commun abstrait appelé `Element`, qui possède trois sous-classes abstraites, correspondant aux trois couches de notre modèle, respectivement `LexicalElement`, `SyntacticElement` et `AggregationElement`. Cette hiérarchie de haut niveau est représentée sur le diagramme de classes de la fig. 1.

Les classes concrètes sont alors dérivées de l'une des classes abstraites, en fonction de la couche à laquelle elles appartiennent. Nous allons passer en revue les classes qui ont

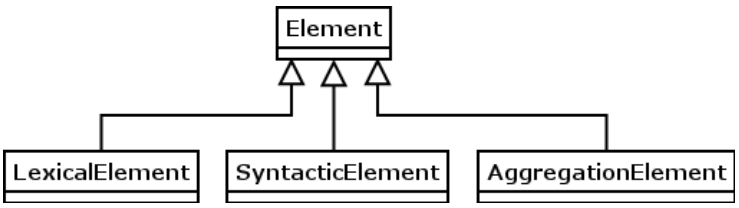

FIG. 1 – Le sommet de notre hiérarchie de classes

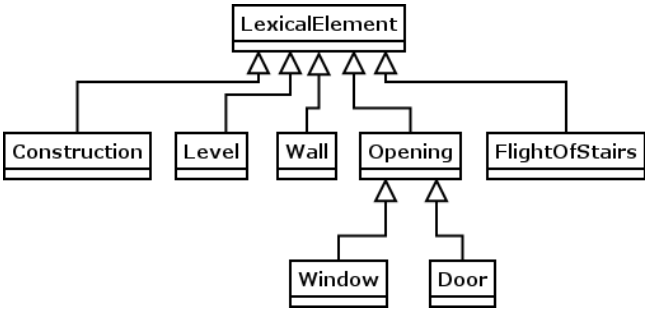

FIG. 2 – Hiérarchie des concepts lexicaux

été définies jusqu'ici – sachant que le nombre de classes est appelé à évoluer de manière à supporter de plus en plus de styles architecturaux réels.

1.2.1 Concepts lexicaux

Les éléments lexicaux sont des éléments architecturaux concrets, positionnés par leurs coordonnées. Chaque élément possède un repère cartésien dans lequel ses sous-éléments sont décrits.

De plus, les éléments lexicaux sont décrits par un ensemble d'attributs physiques, comme leurs coordonnées géométriques ou des attributs de taille.

Voyons les concepts de la fig. 2 :

- une construction (`Construction`) représente le point autour duquel est construit un bâtiment. En pratique, tous les autres éléments du bâtiment seront décrits par rapport au repère associé à ce point ;
- un niveau (`Level`) est une surface plane sur laquelle on peut marcher ;
- un mur (`Wall`) a la signification classique de structure verticale en dur ;
- la classe abstraite des ouvertures dans les murs (`Opening`) possède deux sous-classes, fenêtre (`Window`) et porte (`Door`), qui ont leur signification classique ;
- une volée de marches (`FlightOfStairs`) est un ensemble de marches qui partagent des caractéristiques communes, comme la hauteur et la largeur des marches. Ce n'est pas un escalier, car un escalier peut être composé de plusieurs volées de marches et de paliers.

1.2.2 Concepts syntaxiques

Les éléments syntaxiques sont définis par l'association de plusieurs éléments lexicaux : ce sont des concepts abstraits et non physiques. Ainsi, contrairement aux éléments lexicaux, les éléments syntaxiques n'ont pas d'attributs physiques propres : ils sont décrits par un ensemble de références vers des éléments lexicaux.

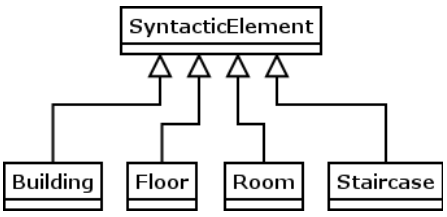

FIG. 3 – Hiérarchie des concepts syntaxiques

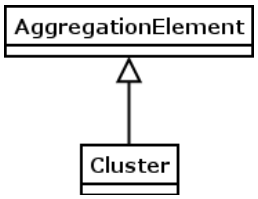

FIG. 4 – Hiérarchie des concepts d'agrégation

Par exemple, une pièce est définie par ses murs (et en pratique, une instance de fenêtre contient des références vers chacun de ses murs).

Les éléments syntaxiques qui ont été définis sont les suivants (cf. fig. 3) :

- un bâtiment (`Building`) est un édifice. Il est associé à une construction (`Construction`) ;
- un étage (`Floor`) peut être composé de plusieurs niveaux (`Level`). Ces niveaux peuvent avoir des altitudes légèrement différentes, et être reliés par des volées de marches, à condition que l'ensemble ait la signification d'un seul étage dans le contexte du bâtiment en cours de description ;
- une pièce (`Room`) est un espace entouré de murs (`Wall`). Il doit y avoir au moins trois murs pour former une pièce ;
- un escalier (`Staircase`) permet de relier deux étages (`Floor`). Il est composé de plusieurs volées de marches, et éventuellement de niveaux qui représentent des paliers.

1.2.3 Concepts d'agrégation

Les concepts d'agrégation permettent de modéliser des structures de plus haut niveau, composées de plusieurs éléments syntaxiques.

Jusqu'à présent, un seul concept a été défini dans cette hiérarchie (fig. 4) : un agrégat (`Cluster`) permet de rassembler plusieurs pièces et étages. Dans ces conditions, un agrégat peut porter de l'information sémantique (voir la section 3) commune à plusieurs éléments syntaxiques. Pour poursuivre l'exemple ci-dessus, le nom et le rôle du "service des ventes" peuvent être associés à l'agrégat qui représente ce service.

Voyons maintenant comment on peut organiser les concepts de cette taxonomie pour construire des descriptions.

1.3 Construction d'une description

Une description suit une structure en arbre. Un tel arbre possède trois sous-arbres, chacun correspondant à une couche de notre modèle.

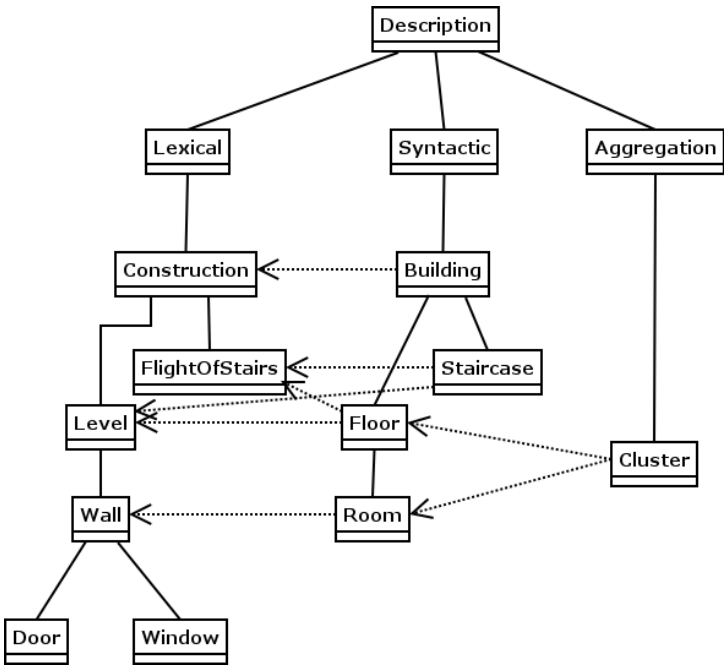

FIG. 5 – Les types de liaisons utilisées dans notre modèle : les traits pleins verticaux représentent les liaisons d'*imbrication*, tandis que les lignes pointillées horizontales représentent les liaisons de *composition*

À l'intérieur d'un sous-arbre donné, les liaisons "verticales" représent l'*imbrication* logique entre éléments : par exemple, une fenêtre (`Window`) est incluse à l'intérieur d'un mur (`Wall`). De la même façon, un mur est imbriqué dans un niveau (`Level`).

Les éléments d'un sous-arbre sont liés à certains éléments de l'arbre frère de gauche (liaisons "horizontales"). Ces liaisons possèdent une signification de *composition*. Les éléments du sous-arbre numéro $n+1$ sont *composés*, c'est-à-dire *définis* par des éléments du sous-arbre numéro n.

La cardinalité des liaisons horizontales est en général multiple, dans le sens qu'un élement de la couche $n+1$ peut être lié à un nombre arbitraire d'élements de la couche n. Cependant, dans certains cas, il existe des restrictions : par exemple, nous avons dit plus haut qu'une pièce doit être liée à au moins trois murs.

2 Expression de notre modèle en XML

2.1 Cahier des charges

Nous venons de décrire un modèle abstrait qui nous permet de réprésenter les architectures de bâtiments. Il s'agit maintenant de lui faire correspondre une représentation concrète en XML. Dans ce but, nous pouvons dresser la liste suivante de contraintes, qui portent sur le modèle de représentation concrète choisi :

1. il doit permettre l'héritage entre concepts, car nous utilisons une description orientée objets ;
2. l'imbrication des instances de classes doit être gérée, car toutes les liaisons verticales modélisent des relations de ce genre ;

3. pour représenter les liaisons horizontales, il doit être possible de relier des instances de classes d'une autre façon que par les relations d'imbrication.

2.2 RDF, XML+DTD ou XML+Schema ?

Actuellement, il existe principalement trois solutions technologiques pour exprimer notre modèle en XML. Analysons les avantages et les inconvénients de chacune d'entre elles.

2.2.1 RDF : Resource Description Framework

RDF [3] [4] est un modèle sous forme de graphe qui permet de décrire des relations entre objets. De plus, les concepts peuvent être définis à l'aide des langages DAML+OIL [5] (Darpa Agent Markup Language, Ontology Inference Layer) ou OWL [6] (Web Ontology Language), qui sont eux-mêmes basés sur RDF.

RDF est juste un modèle abstrait, donc indépendant du mode de représentation, mais il existe une façon standard de l'exprimer à l'aide d'une syntaxe XML [7]. RDF pourrait correspondre à nos besoins car :

1. DAML+OIL et OWL permettent l'héritage : ce sont bien des modèles à objets ;
2. l'imbrication d'instances de classes peut être représentée par des relations entre objets, du type `imbriquéDans` ;
3. toutes les liaison sont possibles, car RDF est un modèle sous forme de graphe.

Mais la polyvalence de RDF a un inconvénient majeur pour notre utilisation : les fichiers générés sont très complexes, et à peu près inéditables à la main.

2.2.2 XML + une définition de type de document (DTD)

C'est la façon standard de définir des modèles XML. Le langage utilisé dans les DTD est très simple, et bien adapté à sa prise en main par un utilisateur. Le façon de procéder la plus évidente est d'associer des éléments XML aux classes. L'imbrication d'instances (contrainte n°2) correspond à l'imbrication d'éléments XML. La contrainte n°3 est satisfaite grâce à l'utilisation d'identificateurs et de références (types `ID` and `IDREF`). Mais il n'y a pas de solution à la contrainte n°1 : avec une DTD, on ne peut pas simplement spécifier des relations d'héritage entre éléments.

2.2.3 XML + un schéma XML

XML Schema [8] est le nouveau langage de description de la structure de documents XML. Son usage n'est pas encore très répandu, mais il est destiné à remplacer les DTD.

L'utilisation de XML, avec des schémas, correspond bien à notre cahier des charges :

1. le langage XML Schema permet la définition de types de données, et il est possible d'"étendre" des types de données pour en créer de nouveaux, ce qui répond à nos besoins d'héritage ;
2. comme précédemment, les relations d'imbrication sont simplement représentées par l'imbrication d'éléments XML ;

3. XML Schema fournit une méthode similaire aux attributs `ID/IDREF` des DTD pour spécifier nos liaisons horizontales. Cependant, les schémas vont plus loin : il est possible par exemple de spécifier le type d'éléments vers lesquels on peut faire des références.

De plus, l'utilisation d'un schéma possède un avantage majeur sur les deux autres méthodes : elle permet la définition de types de données précis pour les attributs, alors que les autres modèles permettent seulement de définir des attributs du type "chaîne de caractères". Avec ce genre de typage fort, on peut forcer les documents instances à respecter une certaine forme de représentation des données, d'où moins de risques de mauvaise interprétation.

Pour toutes ces raisons, nous avons choisi d'utiliser le langage XML, avec un schéma : RDF certes est polyvalent, mais bien trop complexe pour notre application simple, tandis que les DTD n'offrent pas assez de possibilités.

2.3 Structure du schéma

Chaque classe de notre modèle est associée à un type de données dans le schéma. Et à leur tour, ces types de données correspondent à des éléments XML.

Pour spécifier les coordonnées géométriques des éléments lexicaux, nous définissons un type de données simple qui permet l'utilisation d'un nombre suivi d'une unité du système métrique. Ainsi, le modèle donne à l'utilisateur un maximum de flexibilité (il permet de choisir parmi différentes unités), tout en assurant une interprétation non ambigüe des valeurs.

3 Perspectives : intégration de données sémantiques

Nous venons d'expliquer comment décrire l'architecture de bâtiments avec XML. Ainsi, notre appareil devrait être capables de savoir où il se trouve, à l'intérieur d'une description donnée.

Mais ceci n'est pas suffisant : nous devons aussi être capable de fournir à l'utilisateur des *données sémantiques*, c'est-à-dire des données porteuses de sens pour l'utilisateur. Par exemple, quand l'appareil se rend compte qu'il est dans une certaine pièce, il devrait être capable de dire à l'utilisateur le nom de la personne qui y travaille, les consignes de sécurité, etc.

Ainsi, nous devons étiqueter tous les objets importants de notre description avec des données sémantiques. Mais nous devons définir un moyen de les représenter. La façon de procéder la plus simple est d'utiliser un élément XML spécial, par exemple `semantic`, qui pourrait être contenu dans n'importe-quel élément.

Mais quelle forme prendront les données que nous allons associer à nos objets architecturaux ? Bien entendu, nous pourrions utiliser de simples descriptions textuelles, comme par exemple :

```
<semantic>
  <text>
    #210 - Salle de réunion
  </text>
</semantic>
```

Mais, encore une fois, ceci n'est pas suffisant : seul un utilisateur humain peut comprendre ce que signifie ce genre de phrases. Une machine sera complètement incapable de traiter cette information. Ainsi, nous devons définir un formalisme pour donner un *sens* compréhensible par les machines à nos informations sémantiques.

Par exemple, les numéros de bureaux, les occupants des bureaux, etc. devraient être balisés de façon adaptée, avec des étiquettes comme `officeNumber` ou `officeOwner`.

En fait, nous devons définir une taxonomie complète de concepts à utiliser pour représenter les données sémantiques associées aux structures architecturales. Ceci se situe bien au-delà de l'objectif de cet article, et sera le sujet de travaux de recherche à venir.

Conclusion

Nous avons défini un modèle de description d'environnements architecturaux. Nous avons montré comment exprimer ce modèle en utilisant XML et les schémas XML. Ce modèle sera intégré dans notre appareil d'aide au déplacement des non-voyants. L'appareil sera ainsi capable de déterminer sa position dans un environnement donné.

Il reste à réaliser une implémentation test de ce modèle : nous devons concevoir un programme capable d'exploiter ces descriptions d'architecture, de façon à tester les algorithmes de positionnement correspondants. De plus, nous devons étendre notre taxonomie de concepts, de façon à ce qu'elle soit suffisamment riche pour représenter un grand nombre d'architectures réelles. Enfin, nous devrons définir une méthode précise de description des données sémantiques qui seront comuniquées à l'utilisateur.

Références

[1] Farcy, R., Bellik, Y. : Locomotion Assistance for the Blind. Universal Access and Assistive Technology. Keates, S., Langdom, P., Clarkson, P.J., Robinson, P. (Eds.), Springer (2002) 277–284

[2] Bellik, Y., Farcy,R. : Comparison of Various Interface Modalities for a Locomotion Assistance Device. Computers Helping People with Special Needs. Miesenberger, K., Klaus, J., Zagler, W. (Eds.), Springer (2002)

[3] Lassila, O., Swick, R. : Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation (1999). `http ://www.w3.org/TR/REC-rdf-syntax/`

[4] Manola, F., Miller, E. : RDF Primer. W3C Working Draft (2003). `http ://www.w3.org/TR/rdf-primer/`

[5] Connoly, D. et als. DAML+OIL Reference Description. W3C Note (2001). `http ://www.w3.org/TR/daml+oil-reference/`

[6] McGuiness, D. et als. OWL Web Ontology Language Overview W3C Working Draft (2003). `http ://www.w3.org/TR/owl-features/`

[7] Beckett, D. : RDF/XML Syntax Specification (Revised). W3C Working Draft (2003). `http ://www.w3.org/TR/rdf-syntax-grammar/`

[8] Fallside, D. : XML Schema Part 0 : Primer. W3C Recommendation (2001). `http ://www.w3.org/TR/xmlschema-0/`