Rapport Projet Architecture Logicielle

ABALINE Rachid
AHOUNOU Thierry
CHARLEROY André

07 Janvier 2013

Sommaire

1 Introduction **3**

2 Contraintes techniques **4**
2.1 Langage utilisé et paton Visiteur 4
2.2 Le fichier XML 4

3 Patron Composite **7**
3.1 Définition 7
3.2 Intervenants 7
3.3 Utilisation 8
3.4 Diagramme de classe 9

4 Patron Builder **10**
4.1 Définition 10
4.2 Intervenants 10
4.3 Utilisation 11
4.4 Diagramme de classe 11

5 Patron Proxy virtuel **12**
5.1 Définition 12
5.2 Intervenants 12
5.3 Utilisation 13
5.4 Diagramme de classe 13

6 Patron Visiteur **14**
6.1 Définition 14
6.2 Intervenants 14
6.3 Remarques 15
6.4 Utilisation 15
6.5 Diagramme UML de principe 16
6.6 Diagramme de classe 17

7 Le patron MVC **18**
7.1 Définition 18
7.2 Utilisation 18

8 Conclusion **20**

Chapitre 1

Introduction

Dans le cadre de l'unité d'enseignement d'Architecture logicielle, nous devons réaliser un projet qui a pour but de concevoir une interface de configuration par navigateur du micrologiciel d'un appareil. Le cas que nous allons représenter dans ce projet est celui d'un routeur.

Le noyau de l'appareil, qui est repésenter par l'interface IDevice nous est fourni, ainsi que la classe Device, qui implémente l'interface.

Dans un premier temps, nous allons explicité les contraintes texchniques.

Dans un second temps nous allons décrire séparement les différents patrons de conception utilisé pour ce projet.

Dans un troisième temps nous allons décrire le modèle MVC.

Chapitre 2

Contraintes techniques

2.1 Langage utilisé et paton Visiteur

Le langage de programmation utilisé pour ce projet est le langage Java (il s'agit d'un langage orientée objet). C'est le langage que nous avons utilisé tout au long de l'unité d'enseignement et il nous permet d'utiliser les patrons de conception.
L'autre contrainte fut d'utiliser le patron VISITEUR qui n'a pas était vu en cours, afin de rester le plus proche possible de la repésentation mémoire de la configuration et du modèle MVC. Ce patron sera détaillé plus tard dans le rapport.

2.2 Le fichier XML

Le fichier XML nous était donné afin de pouvoir réaliser le projet. Il représente la structure de configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE category SYSTEM "config.dtd">
-<category name="home">
-<info name="hardware description">
```

```
<infoline title="hardware" desc="SysLink router"/>
<infoline title="type" desc="network router"/>
<infoline title="version" desc="2.01b"/>
</info>
-<category name="Edit settings">
-<category name="lan">
-<module name="Ethernet 1">
<checkfield name="enabled" settingkey="lan.eth1.enabled"/>
<checkfield name="negotiation" settingkey="lan.eth1.negociation"/>
<textfield name="mac adresse" settingkey="lan.eth1.mac"/>
</module>
-<module name="Ethernet 2">
<checkfield name="enabled" settingkey="lan.eth2.enabled"/>
<checkfield name="negotiation" settingkey="lan.eth2.negociation"/>
<textfield name="mac adresse" settingkey="lan.eth2.mac"/>
</module>
-<module name="Ethernet 3">
<checkfield name="enabled" settingkey="lan.eth3.enabled"/>
<checkfield name="negotiation" settingkey="lan.eth3.negociation"/>
<textfield name="mac adresse" settingkey="lan.eth3.mac"/>
</module>
-<module name="Ethernet 4">
<checkfield name="enabled" settingkey="lan.eth4.enabled"/>
<checkfield name="negotiation" settingkey="lan.eth4.negociation"/>
<textfield name="mac adresse" settingkey="lan.eth4.mac"/>
</module>
-<info name="help">
<infoline title="Settings" desc="This menu can be used to edit the ethernet configuration of
</info>
</category>
-<category name="wifi">
-<module name="Antenna">
<checkfield name="right enabled" settingkey="wifi.rant.enabled"/>
<checkfield name="left enabled" settingkey="wifi.lant.enabled"/>
-<choicefield name="channel" settingkey="wifi.channel">
<choicevalue name="1 - 2412 Mhz" value="1"/>
<choicevalue name="2 - 2417 Mhz" value="2"/>
<choicevalue name="3 - 2422 Mhz" value="3"/>
<choicevalue name="4 - 2427 Mhz" value="4"/>
</choicefield> </module> -<module name="Access Point">
-<choicefield name="type" settingkey="accesspoint.type">
<choicevalue name="Infrastructure" value="1"/>
<choicevalue name="adhoc" value="2"/> </choicefield>
<checkfield name="routing" settingkey="accesspoint.routing.enabled"/>
<textfield name="left enabled" settingkey="accesspoint.gateway"/>
<textfield name="SSID" settingkey="wifi.ssid"/>
```

```
</module>
-<info name="help">
<infoline title="Settings" desc="This menu can be used to edit the internal wireless setting
</info>
</category>
-<info name="help">
<infoline title="Settings" desc="This menu can be used to edit the internal settings of the
</info>
</category>
</category>
```

Il se compose de deux types d'élements :

Element noeud : categorie, info, module, choicefield. Ces éléments peuvent contenir plusieurs autres éléments.

Element feuille : checkfield, textfield, choiceValue, infoLine. Chacun des ces élements reprèsentent un objet au sein d enotre projet, qui sera caractérisé par des attributs et des mèthodes.

Chapitre 3

Patron Composite

3.1 Définition

Le patron COMPOSITE uniforme le comportement d'objets et de lots d'objets.

3.2 Intervenants

Il y a trois intervenants dans le patron COMPOSITE :

Composant : il s'agit d'une classe abstraite ou interface qui introduit le comportement des objets de la composition, implante les méthodes communes et introduit la signature des méthodes qui gèrent la composition en y ajoutant ou en supprimant des composants.

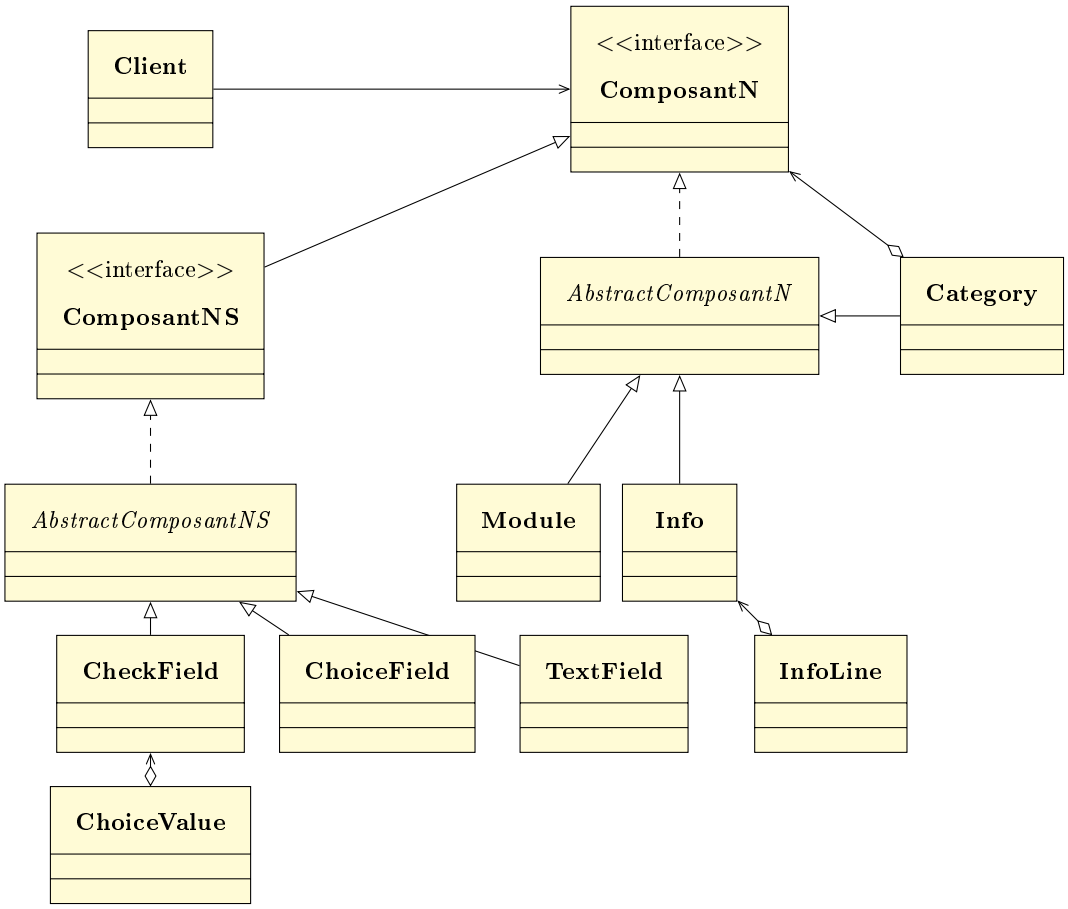
Feuille : Il s'agit de la classe concrète qui décrit les feuilles de la composition (une feuille ne possède pas de composants).

Composite : Il s'agit d'une classe concrète, qui décrit les objets composés de la hiérarchie. Cette classe possède unee association d'agrégagtion avec la classe ou interface Composant.

3.3 Utilisation

Lorsque nous avons étudié la structure du fichier XML, il nous a paru évident d'utiliser ce patron car les catégories correspondent à des sections de menu pouvant regrouper d'autres catégories, donc joue le rôle de Composite. Les infos et les modules sont des sections qui ne peuvent pas avoir d'autre contenu donc joue le rôle de Feuille. Cela nous permet donc de grouper des sections uniques ou des sections contenant d'autres catégories en accord avec le fichier XML et d'uniformiser leur traitement. Le client lui ne se rend pas compte entre les feuilles et la composition.

3.4 Diagramme de classe



L'interface ComposantN représente les éléments qui ont seulement l'attribut name.

L'interface ComposantNS représente les éléments qui ont comme attribut name et settingKey.

Chapitre 4

Patron Builder

4.1 Définition

Lobjectif du patron BUILDER est de déplacer la logique de construction d'un objet en dehors de la classe à instancier.

4.2 Intervenants

Les différents intervenants du patron sont :

Director : Demande la construction dun objet de type Product par lintermédiaire de linterface (ou classe abstraite) Builder.

Builder : spécifie linterface (ou classe abstraite) déclarant les méthodes de création des parties dun objet de type Product.

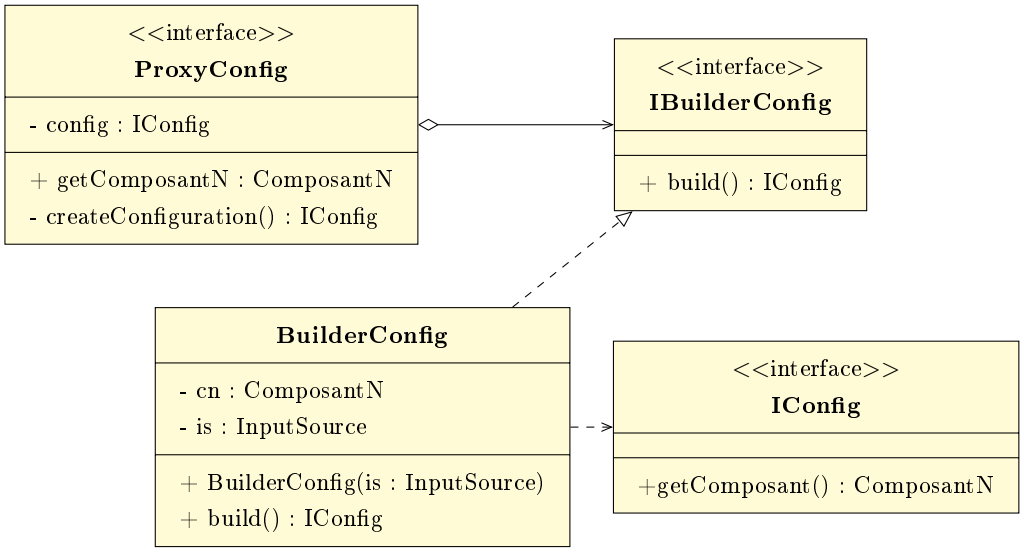
Concretebuilder : Réalise linterface Builder en construisant et assemblant les parties du produit.

Product : Classe ou interface représentant lobjet complexe à construire.

4.3 Utilisation

Nous l'avons utilisé car ce patron nous permetter de récupérer les données contenues dans le fichier XML "config.XML". Le fichier XML contient la structure de l'interface utilisateur qui est en accord avec le fichier DTD "config.DTD" Le patron BUILDER va analyser la cohérence des données avant l'instanciation d'un objet.

4.4 Diagramme de classe



Chapitre 5

Patron Proxy virtuel

5.1 Définition

Lobjectif du patron PROXY VIRTUEL est de fournir un intermédiaire qui permette de contrôler l'accès à un objet. L'objet qui effectue la substitution possède la même interface que le sujet, ce qui rend cette substitution transparente vis-à-vis des clients.

5.2 Intervenants

Les différents intervenants du patron sont :

Sujet : il s'agit de l'interface commine au proxy et au sujet réel.

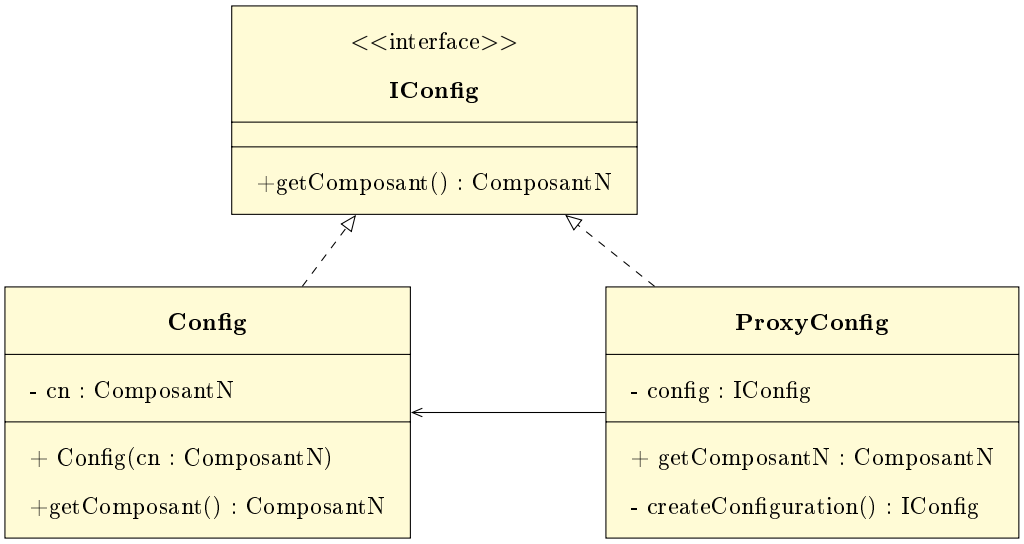
SujetRéel : il s'agite de l'objet que le proxy contrôle et repésente.

Proxy : il s'agit de l'objet qui se substitue au sujet réel. Il possède une interface identique à ce dernier (interface Sujet). Il est chargé de créer et de détruire le sujet réel et de lui déléguer les messages.

5.3 Utilisation

Le patron Proxy que nous avons utilisé joue le rôle de mandataire entre la classe IConfig qui représente les éléments de notre fichier XML sous forme d'objets construit avec le patron BUILDER et notre client.

5.4 Diagramme de classe



Chapitre 6

Patron Visiteur

6.1 Définition

Le pattern VISITEUR construit une opération à réaliser sur les éléments d'un ensemble d'objets. De nouvelles opérations peuvent ainsi être ajoutées sans modifier les classes de ces objets.

6.2 Intervenants

Il y a quatre intervenants dans le patron Visiteur :

Visiteur : Il s'agit de l'interface qui introduit la signature des méthodes qui réalisent une foncitonnlité au sein d'un ensemble de classes. Il existe une méthode par classe qui reçoit comme argument une instance de cette classe

VisiteurConcret : Il s'agit des classes implantant les méthodes qui réalisent la fonctionnalité correspondant à la classe. Cette fonctionnalité est distribuée dans les différents éléments.

Visitable : Il s'agit d'une classe abstraite, classe mère des classes des classes éléments. elle introduit la méthode abstraite *accepteVisiteur* qui accepte un visiteur comme argument.

Élément : Classe implantant la méthode *accepteVisiteur* qui consiste à rappeler le visiteur au travers de la méthode correspondant à la classe.

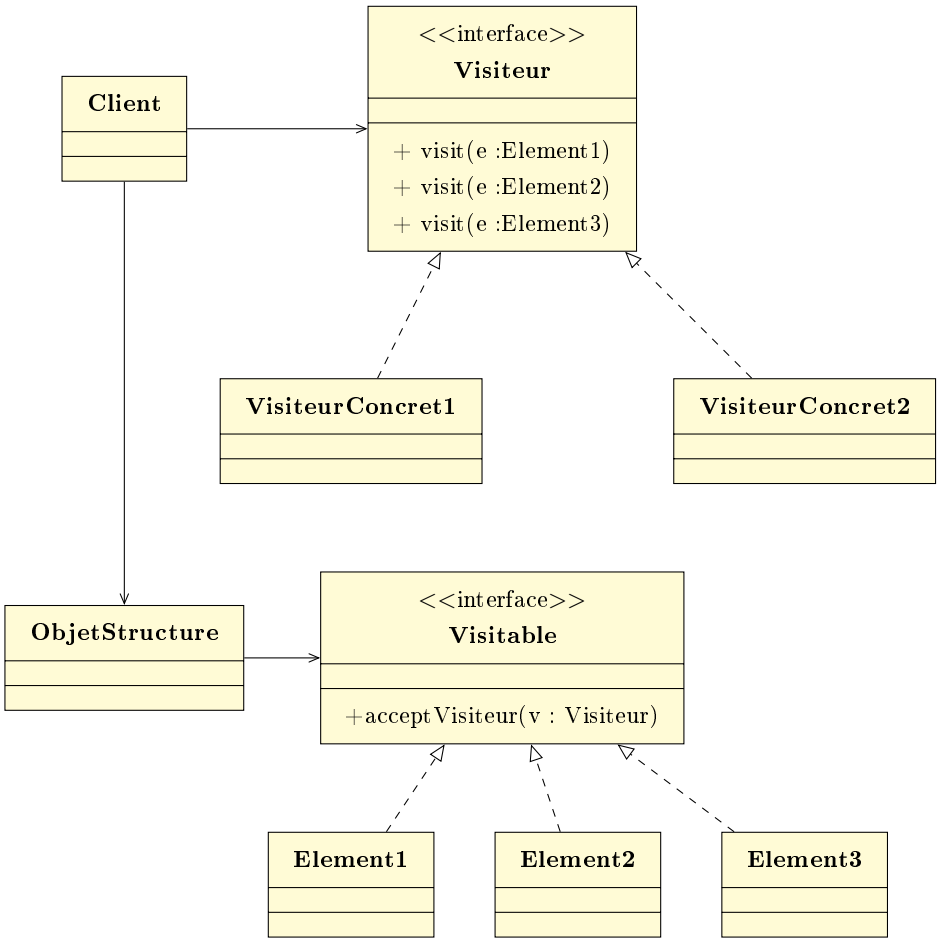
6.3 Remarques

Un client qui utilise un visiteur doit d'abord le créer comme instance de la classe de son choix puis le transmettre comme argument de la méthode *acceptéVisiteur* d'un ensemble d'éléments.
L'élément rappelle la méthode du visiteur qui correspond à sa classe. Il transmet une référence vers lui-même comme argument afin que le visiteur puisse accéder à sa structure interne.
Il faut utiliser ce patron que dans le cas où le nombre de fonctionnalités reste faibles, sinon il y aura beaucoup de méthodes dans les classes, ce qui deviendra difficile à appréhender et à maintenir.

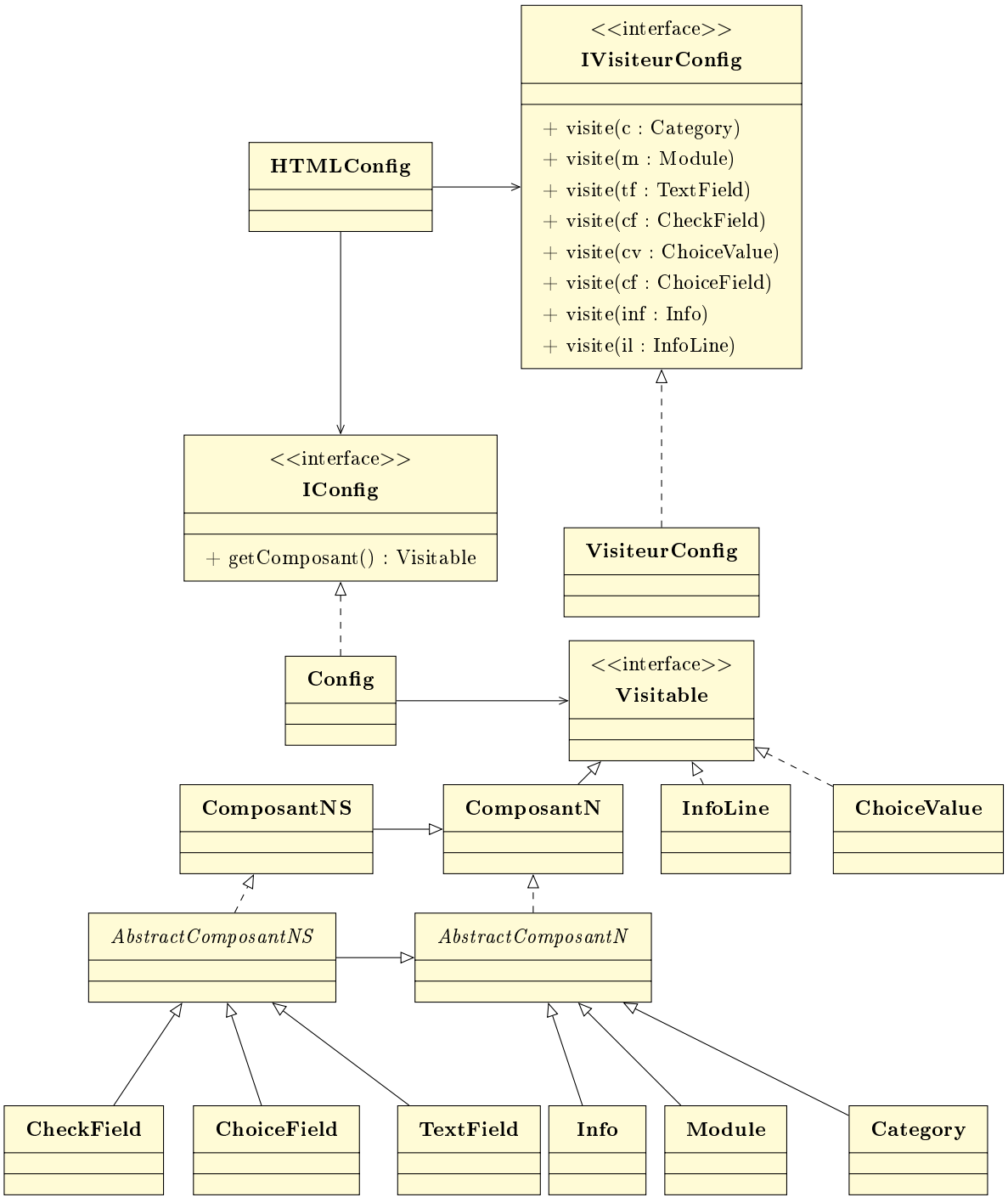
6.4 Utilisation

Le patron VISITEUR nous permet de gérer un traitement special pour chaque objet visitable, chaque objet représente un élément de notre fichier XML et accepte un visiteur qui contient une méthode spécifiques "visite" qui traitre l'affichage ce ce dernier.

6.5 Diagramme UML de principe



6.6 Diagramme de classe



Chapitre 7

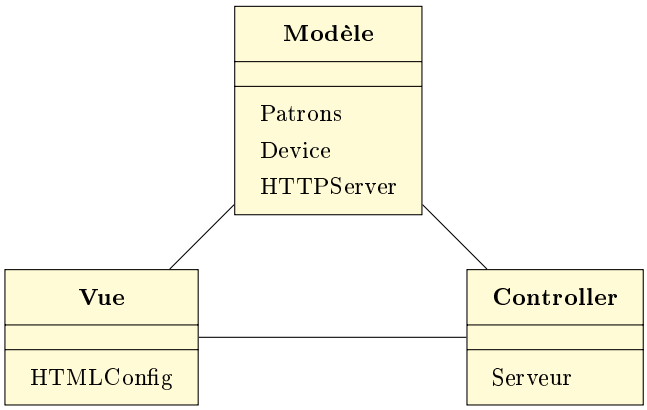
Le patron MVC

7.1 Définition

Le patron Modèle-vue-contrôleur (en abrégé MVC, de l'anglais Model-View-Controller), tout comme les patrons Modèle-vue-présentation ou Présentation, abstraction, contrôle, est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective.

7.2 Utilisation

Pour notre projet, nous avons découper ainsi :


Modèle
Patrons
Device
HTTPServer
Vue
HTMLConfig
Controller
Serveur

Chapitre 8

Conclusion

L'objectif de ce projet était le développement d'une interface de configuration par navigatuer du micrologiciel d'un routeur. Nous avons utiliser différents patrons de conception afin de concevoir une application flexibles et réutilisable permettant de configurer un appareil. Pour cela il était important de coder le plus proprement possible. Donc l'utilisation de patrons de conception nous permet de bien structurer notre projet.