[Projet n°2 : Objecteering]

[M2 MIAGE - Florent DAUDIN
Frédéric MADSEN]

[2009]

[But du projet : Utiliser le logiciel Objecteering, et faire une petite application.]

UNIVERSITÉ D'EVRY
VAL D'ESSONNE

Boulevard François Mitterrand
91025 Evry CEDEX

[Encadré par :
Monsieur Henry BOCCON-GIBOD
& Monsieur Vincent GODEFROY]

miage EVRY
Méthodes Informatiques Appliquées

[Sommaire]

Chapitre 1. Cahier des charges

Section 1.01 Le but du projet

Le but du projet est d'utiliser le logiciel Objecteering, et faire une petite application.

On portera par ailleurs un regard critique sur la pertinence opérationnelle de la démarche MDA dans le cadre de l'utilisation de ce logiciel.

Section 1.02 Les étapes du projet

(a) Etudier Objecteering

Dans un premier temps, nous devons étudier le logiciel Objecteering : son installation, ses pré-requis et son fonctionnement.

(b) Utiliser Objecteering

Dans un second temps, nous allons devoir utiliser le logiciel Objecteering afin d'établir une démonstration de ses capabilités.

Section 1.03 L'application Objecteering

Pour créer cette application, nous allons devoir utiliser nos connaissances acquises en UML et en SOA.

Section 1.04 Le rapport

Le rapport final comportera la présentation du logiciel Objecteering et la présentation de l'application, ainsi que quelques tutoriaux sur des aspects importants via des animations Flash réalisées à l'aide de l'outil Wink (http://www.debugmode.com/wink/).

Chapitre 2. Introduction

Section 2.01 MDA

Nouvelle approche d'ingénierie logicielle élaborée par l'OMG, MDA (Model Driven Architecture) prône l'utilisation systématique de modèles dans toutes les phases du cycle de vie des applications informatiques, de l'analyse et conception à la génération de code et au déploiement.

Pour une meilleure productivité et une plus grande pérennité des développements, la logique métier d'une application est modélisée indépendamment de toute plate-forme d'exécution, un mécanisme de transformation de modèles permettant ensuite de produire automatiquement les modèles et le code spécifiques à chaque plate-forme.

Section 2.02 Objecteering

Il s'agit d'un atelier de génie logiciel fondé initialement sur la méthode classe/relation qui a par la suite intégré UML et qui a été le premier à intégrer le concept de profil UML.

Objecteering est un outil qui englobe l'ensemble des phases d'un projet, de la gestion des exigences en passant par toutes les phases de modélisation jusqu'à la génération du code et les plans de test, et qui fait naturellement intervenir plusieurs métiers.

De part cette spécificité nous nous sommes concentré sur certaines phases qui nous semblaient accessibles au vu de nos connaissances actuelles.

Chapitre 3. Présentation du logiciel Objecteering

Section 3.01 Sa présentation

Objecteering est un outil de modélisation UML avancé, développé et commercialisé par Objecteering Software, une filiale de Softeam.

Objecteering offre un support complet de l'UML2. Il est bien connu pour le support étendu MDA qu'il fournit, ses fonctionnalités de travail en groupe avancées, ses générateurs de code, et ses paradigmes de synchronisation code-modèle.

Section 3.02 Son téléchargement

➔ Pour télécharger une démo du logiciel Objecteering il faut aller sur le site : http://www.objecteering.com/

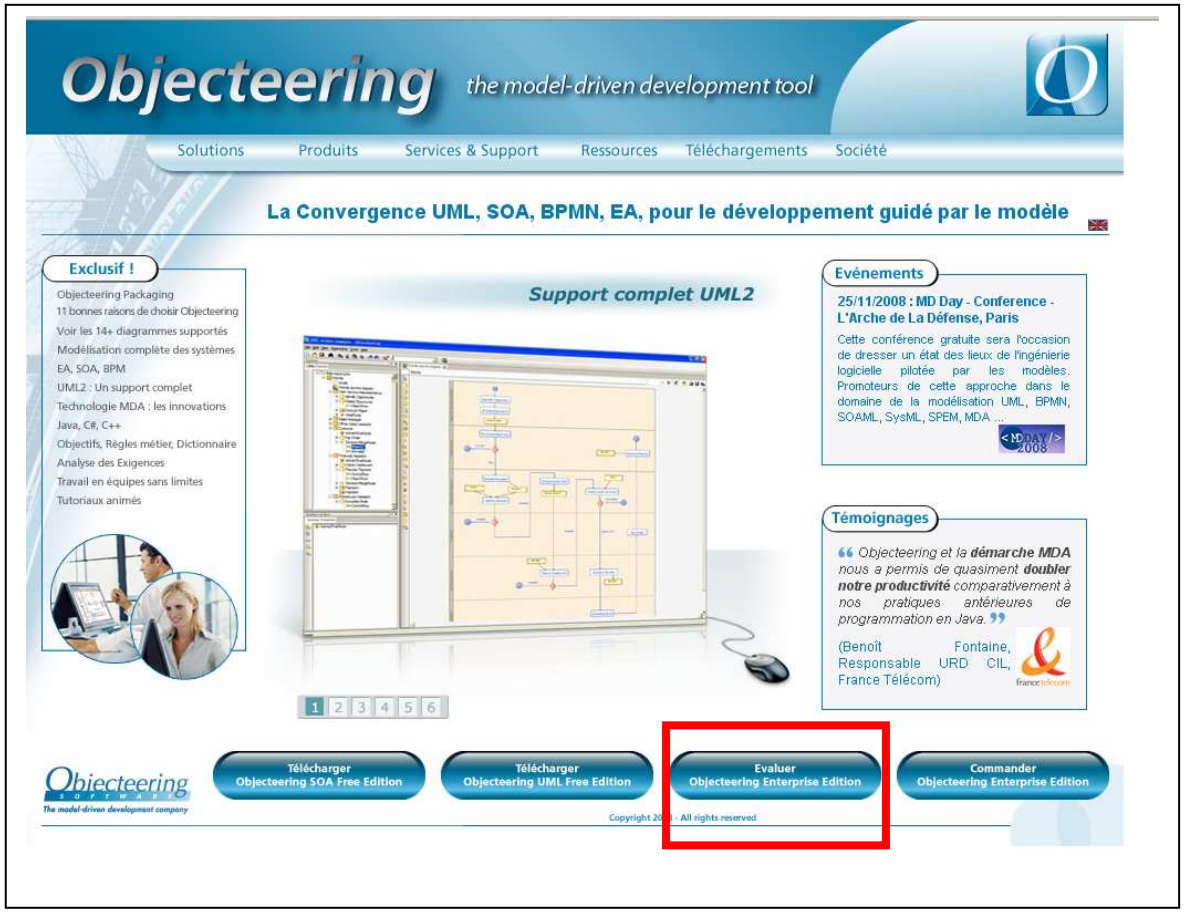


Fig. 1

➔ Nous avons choisi la version « Evaluate Objecteering Enterprise Edition » Nous devions remplir un formulaire pour avoir accès au téléchargement du

logiciel.
Ce formulaire nous demandait le host ID de notre machine. Nous devions télécharger un logiciel permettant de nous donner ce numéro.

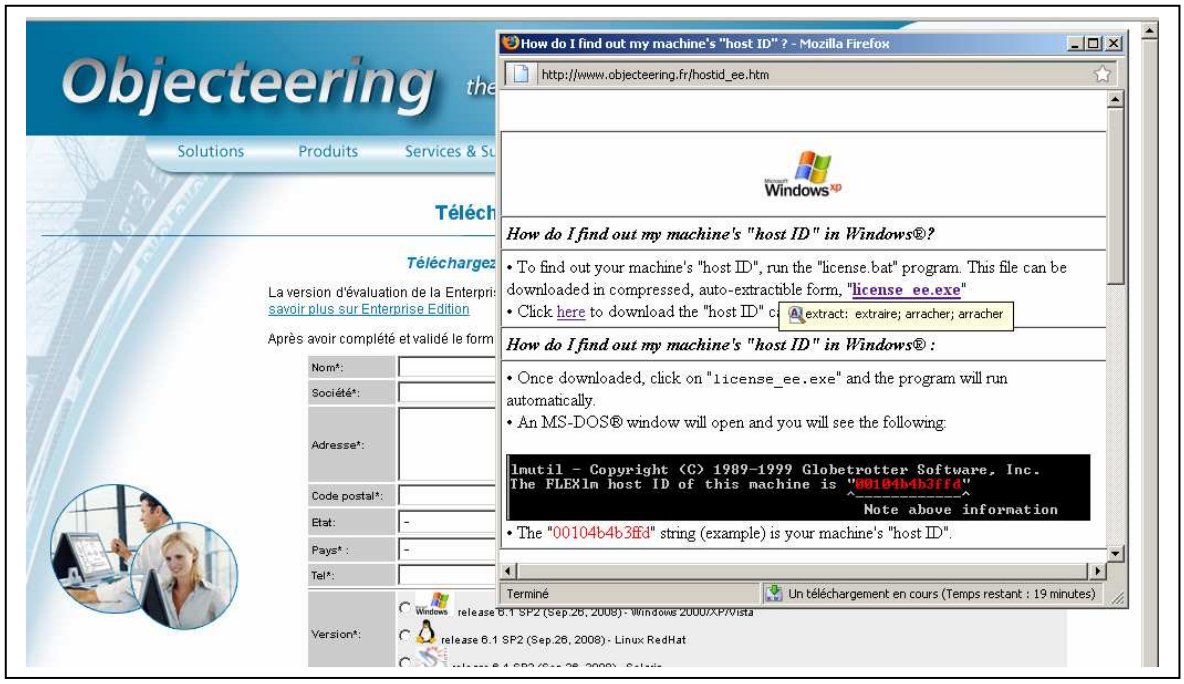


Fig. 2

➔ Après avoir rempli le formulaire, nous avions accès au logiciel.
Un email nous a été adressé avec un fichier : « license.lic »

Fig. 3

Section 3.03 Son installation

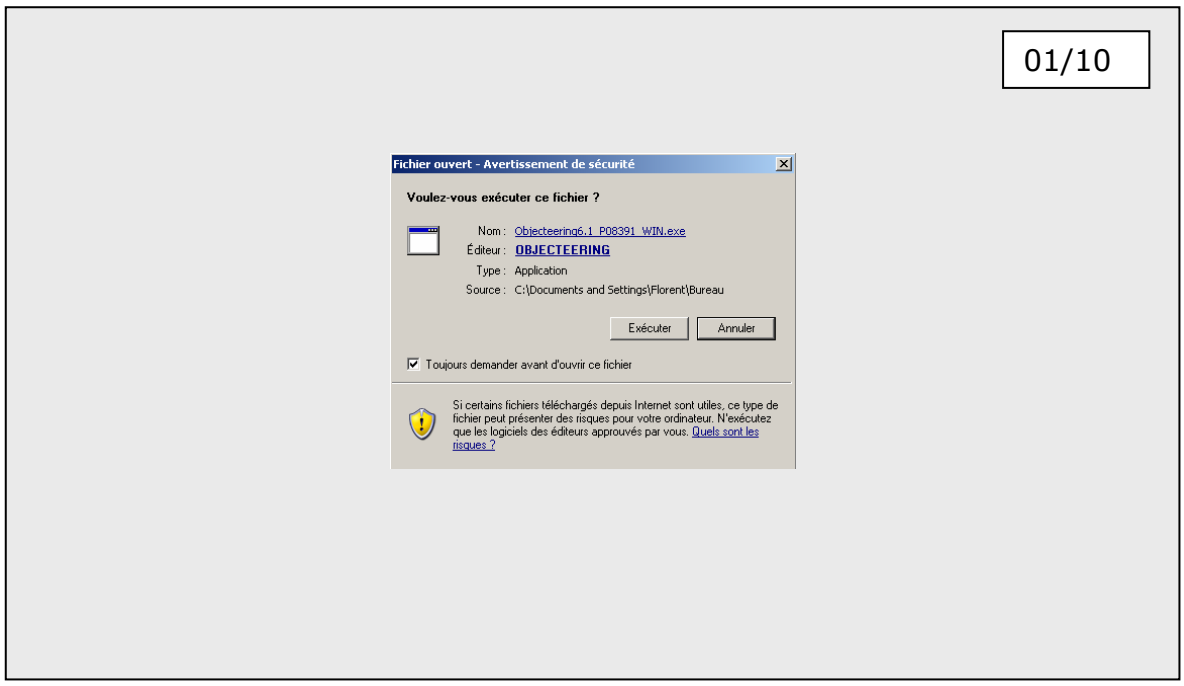


Fig. 4

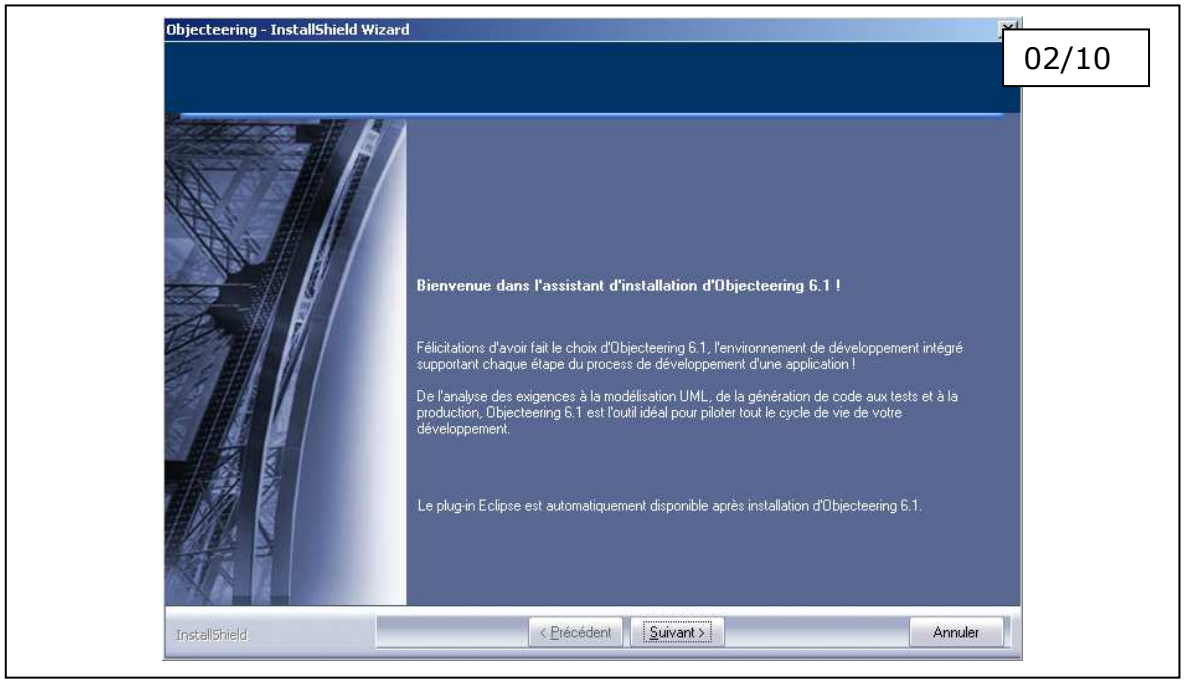


Fig. 5


Objecteering - InstallShield Wizard
Contrat de Licence d'Utilisateur Final.
03/10
XV. DISPOSITIONS DIVERSES
Les documents constitutifs du contrat sont les bordereaux de livraison et les conditions générales. Ils expriment l'intégralité de l'accord entre les parties et ils se substituent à tout autre document, accord, écrit ou verbal, portant sur le même objet et antérieurs au contrat. Toute modification du contrat signé entre les parties ne peut intervenir que par voie d'avenant accepté et signé par le Client et par Objecteering Software. Les présentes conditions générales excluent l'application des éventuelles conditions générales du Client.
XVI. LOI APPLICABLE - JURIDICTION COMPETENCE
Le présent Contrat est soumis, à la loi Française. Tout litige susceptible de s'élever entre les parties à propos de la formation, de l'exécution, ou de l'interprétation du présent contrat sera de la compétence exclusive du Tribunal de Commerce du Siège Social de Objecteering Software.
J'accepte les termes du Contrat de licence
Je n'accepte pas les termes du Contrat de licence
Imprimer
InstallShield
< Précédent
Suivant >
Annuler

Fig. 6


Objecteering - InstallShield Wizard
Informations client
Nom du détenteur de la licence Objecteering 6.1.
04/10
Veuillez donner le nom et la société du détenteur de la licence Objecteering 6.1.
Nom du détenteur :
Boomscud
Nom de la société :
Boomscud

Fig. 7

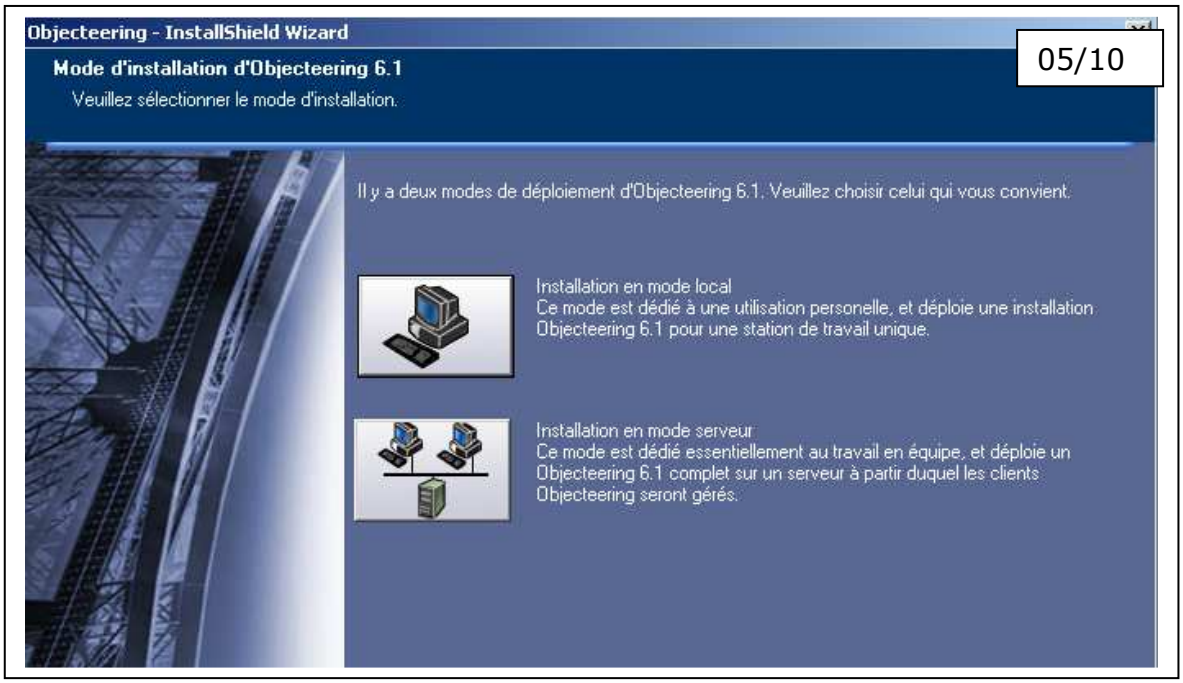

Objecteering - InstallShield Wizard
Mode d'installation d'Objecteering 6.1
Veuillez sélectionner le mode d'installation.
05/10
Il y a deux modes de déploiement d'Objecteering 6.1. Veuillez choisir celui qui vous convient.
Installation en mode local
Ce mode est dédié à une utilisation personelle, et déploie une installation Objecteering 6.1 pour une station de travail unique.
Installation en mode serveur
Ce mode est dédié essentiellement au travail en équipe, et déploie un Objecteering 6.1 complet sur un serveur à partir duquel les clients Objecteering seront gérés.

Fig. 8


Objecteering - InstallShield Wizard
Répertoire d'installation
Veuillez indiquer le dossier où Objecteering 6.1 sera installé.
06/10
Veuillez indiquer le dossier où Objecteering 6.1 sera installé.
Vous pouvez utiliser le dossier par défaut ou en sélectionner un autre en cliquant sur Parcourir.
Espace disque nécessaire (approximatif) : 374 MB
Dossier de destination

Fig. 9

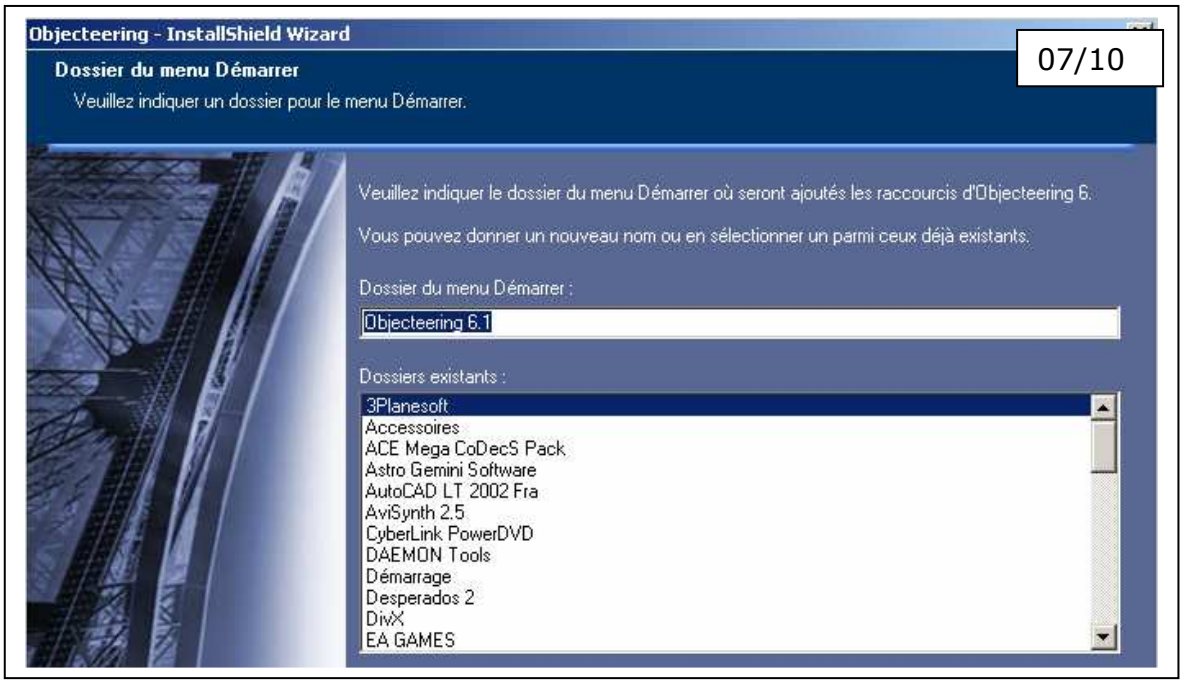

Objecteering - InstallShield Wizard
07/10
Dossier du menu Démarrer
Veuillez indiquer un dossier pour le menu Démarrer.
Veuillez indiquer le dossier du menu Démarrer où seront ajoutés les raccourcis d'Objecteering 6.
Vous pouvez donner un nouveau nom ou en sélectionner un parmi ceux déjà existants.
Dossier du menu Démarrer :
Objecteering 6.1
Dossiers existants :
3Planesoft
Accessoires
ACE Mega CoDecS Pack
Astro Gemini Software
AutoCAD LT 2002 Fra
AviSynth 2.5
CyberLink PowerDVD
DAEMON Tools
Démarrage
Desperados 2
DivX
EA GAMES

Fig. 10

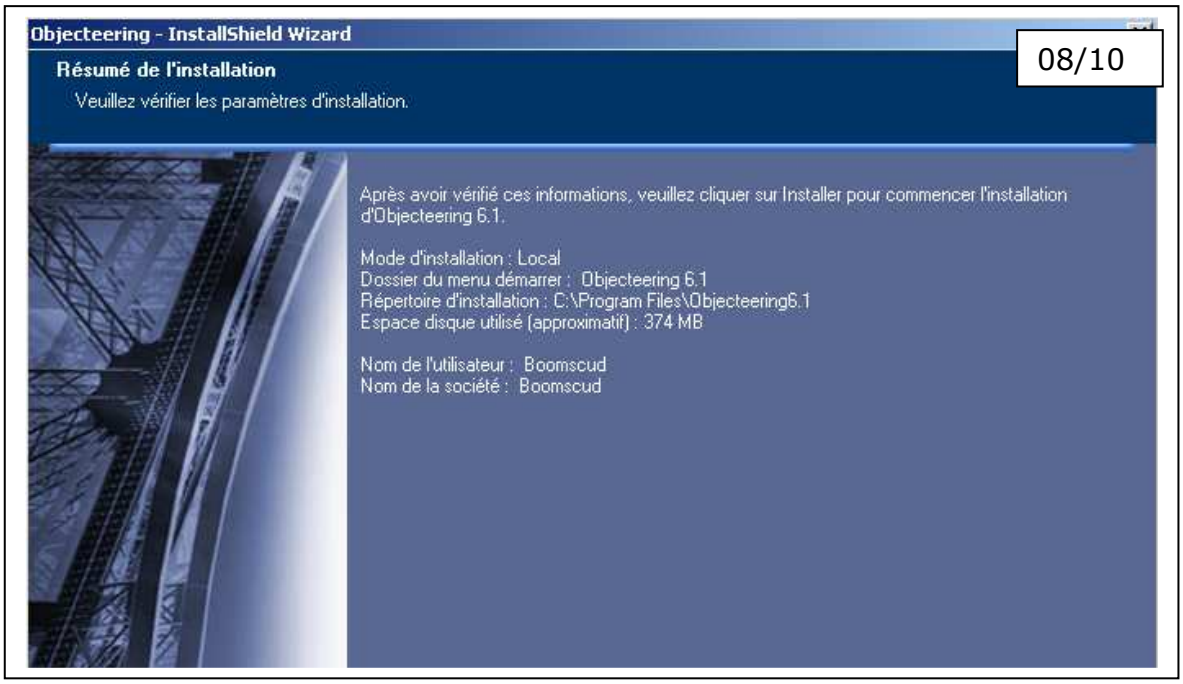

Objecteering - InstallShield Wizard
08/10
Résumé de l'installation
Veuillez vérifier les paramètres d'installation.
Après avoir vérifié ces informations, veuillez cliquer sur Installer pour commencer l'installation d'Objecteering 6.1.
Mode d'installation : Local
Dossier du menu démarrer : Objecteering 6.1
Répertoire d'installation : C:\Program Files\Objecteering6.1
Espace disque utilisé (approximatif) : 374 MB
Nom de l'utilisateur : Boomscud
Nom de la société : Boomscud

Fig. 11

Fig. 12

Fig. 13

➔ Au début de son installation le logiciel demande le fichier : license.lic
On peut utiliser le logiciel pour une durée de 1 mois.
Un email a été adressé à l'administrateur Système d'Evry, Monsieur Djelloul Hanichi afin d'obtenir une licence plus longue.

Section 3.04 Découverte du logiciel Objecteering

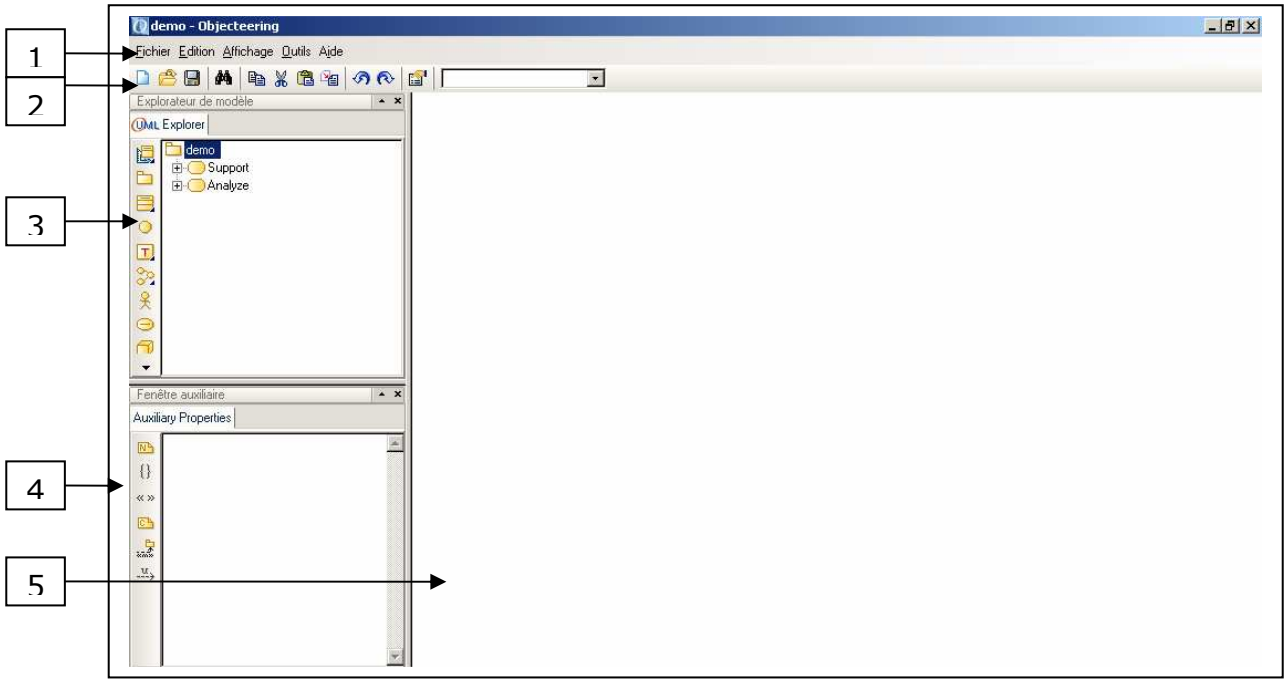


Fig. 14

Légende :

1. Barre de menu
2. Barre d'outils
3. Explorateur principal
4. Editeur de propriétés
5. Editeur graphique

Chapitre 4. Gestion des exigences

Section 4.01 Présentation

La phase amont avant toute modélisation est l'écriture d'un cahier des charges.

Le CDC constitue alors la première étape, non formelle qu'il convient de rendre le plus formel possible afin d'en déduire des processus et règles de gestions.

Les CDC étant généralement rédigés sous Word ou Open Office, Objecteering propose un module qui permet directement au sein de notre document d'annoter ce qu'il conviendra de considérer comme une exigence, un terme du dictionnaire, un objectif ou encore une règle métier.

Une fois notre CDC parcouru et dépouillé du superflu, seule les phrases ayant une connotation propre à définir notre projet son exportées au format XML.

Les objets ainsi créés peuvent ensuite être associés à des éléments de la modélisation, assurant ainsi une traçabilité en amont, bien avant l'apparition de diagrammes UML concernant le projet.

À ce niveau on peut noter qu'une intégration plus profonde de notre éditeur favori (Word) serait la bienvenue, ainsi cela permettrait de prendre directement en compte toute modification de notre cahier des charges.

Malheureusement, seul un raccourci dans l'explorateur de projet d'Objecteering permet d'accéder rapidement au CDC.

Section 4.02 Démonstration

Ces vidéos montrent comment installer le module Scope Manager dans Word, exporter le fichier XML correspondant puis importer le tout dans Objecteering :

- Utilisation du modèle Scope Manager dans :
  - Word
  - Objecteering

Chapitre 5. Notre application

Section 5.01 Thème

Nous allons créer notre application sur le thème d'un site de vente en ligne, un site de e-commerce.

Section 5.02 Diagramme de Classe

(a) Réalisation d'un diagramme de classe

Le diagramme de classes est le point central dans un développement orienté objet. En analyse, il a pour objet de décrire la structure des entités manipulées par les utilisateurs. En conception, le diagramme de classes représente la structure d'un code orienté objet ou, à un niveau de détail plus important, les modules du langage de développement.

(b) Eléments que l'on peut créer dans diagramme de classe

Icône	Description
	Pointeur flèche
	Créer un package
	Créer une classe
	Créer une collaboration
	Créer une interface
	Créer un type de donnée
	Créer une opération
	Créer un attribut
	Créer un port
	Créer une instance
	Créer un lien d'héritage ou une implémentation d'interface
	Créer une association
	Créer un lien
	Créer un import de package ou un import d'élément ou un lien d'utilisation générique.
	Créer un lien de dépendance
	Créer un flux d'information
	Créer un élément d'information
	Créer une note

(c) Explications sur la construction d'un diagramme de classe

Dans un diagramme de classe, nous avons des :

- Classes
    - Une classe représente la description abstraite d'un ensemble d'objets possédant les mêmes caractéristiques. On peut parler de type. Exemples : la classe voiture, la classe personne.
- Objets
    - Un objet est une entité aux frontières bien définies, possédant une identité et encapsulant un état et un comportement. Un objet est une instance (ou occurrence) d'une classe. Exemples : Pascal Roques est un objet instance de la classe Personne. Le livre que vous tenez entre vos mains est une instance de la classe Livre.
- Attributs
    - Un attribut représente un type d'information contenu dans une classe. Exemples : vitesse courante, cylindrée, numéro d'immatriculation, etc. sont des attributs de la classe voiture.
- Associations
    - Une association représente une relation sémantique durable entre deux classes. Exemple : Une personne peut posséder des voitures. La relation possède est une association entre les classes Personne et Voiture.
      Attention : même si le verbe qui nomme une association semble privilégier un sens de lecture, une association entre concepts dans un modèle du domaine est par défaut bidirectionnelle. Donc implicitement, l'exemple précédent inclut également le fait qu'une voiture est possédée par une personne.
- Opérations
    - Une opération représente un élément de comportement (un service) contenu dans une classe.

(d) Notre diagramme de classe

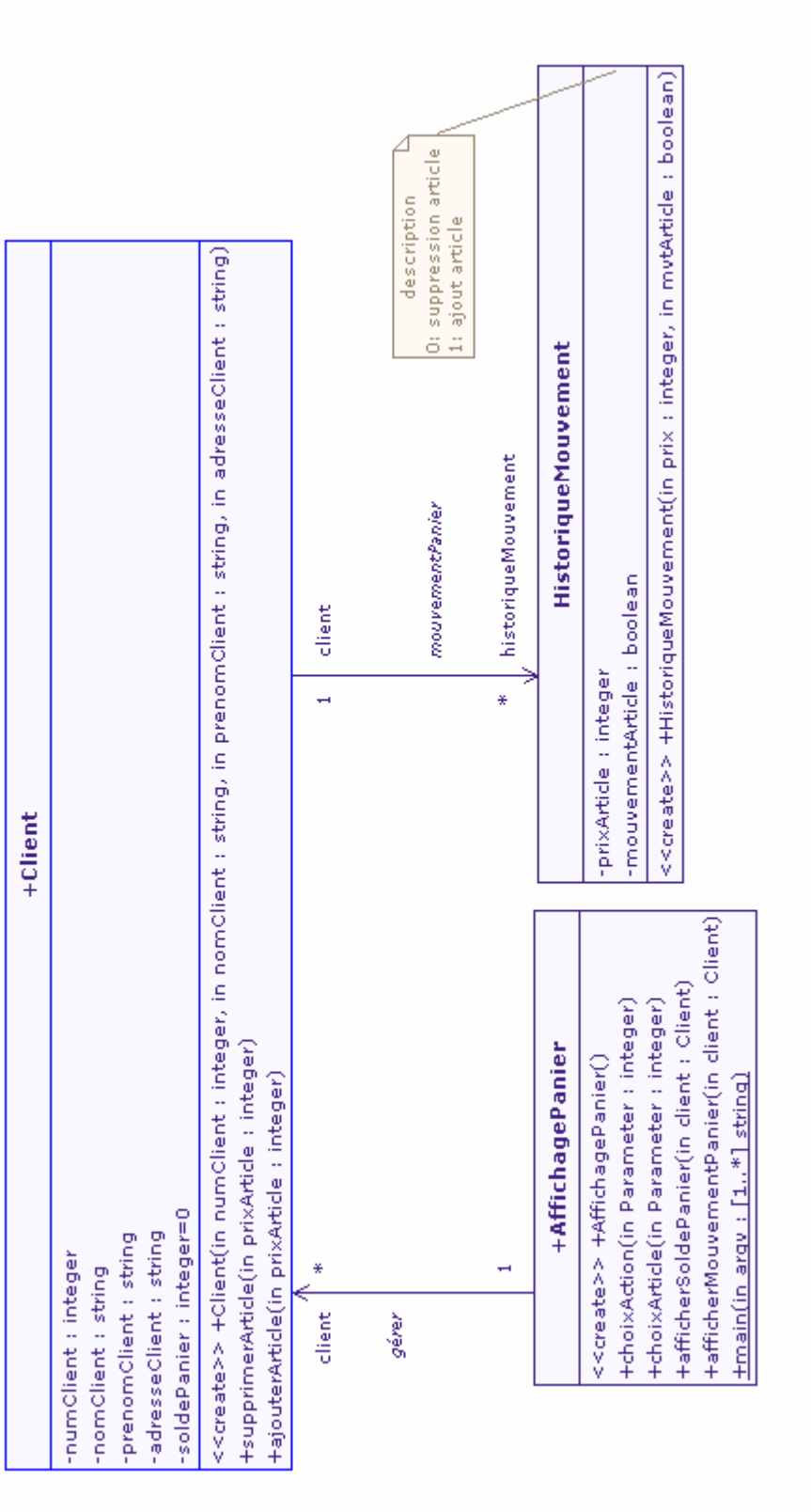


Fig. 15

Section 5.03 Diagramme des cas d'utilisation

(a) Réalisation d'un cas d'utilisation

Un cas d'utilisation comprend :

- Les acteurs,
- Le système,
- Le cas d'utilisation lui-même (fonctionnalité).

Pour utiliser les outils (les icônes), il suffit de cliquer sur l'icône et ensuite de cliquer dans l'éditeur graphique.

Pour personnaliser le texte de chaque élément, il suffit de « double-cliquer » sur celui-ci.

(b) Eléments que l'on peut créer dans un cas d'utilisation

Icône	Description
	Pointeur fléche
	Créer un cas d'utilisation
	Créer un acteur
	Créer un point d'extension
	Créer un lien de communication (une association)
	Créer une relation d'inclusion
	Créer une relation d'extension
	Créer un lien de dépendance
	Créer un lien d'héritage
	Créer un flux d'information
	Créer une note

(c) Explications sur la construction d'un use case

Il existe quatre catégories d'acteurs :

- *Les acteurs principaux* : regroupe les personnes qui utilisent les principales fonctions du systèmes.
- *Les acteurs secondaires* : regroupe les personnes qui effectuent des tâches de maintenance/ d'administration.
- *Le matériel externe* : dispositifs matériels utilisés par le système (ex : imprimante).
- *Les autres systèmes* : systèmes avec lesquels le système doit interagir, mais qui ne font pas partie du domaine d'application.

Les différents types de relations sont les suivantes :

- *Relation de communication* : un acteur déclenche un cas d'utilisation.
- *Relation d'utilisation* : un cas d'utilisation A utilise un cas d'utilisation B.
- *Relation d'extension* : un cas d'utilisation A (source) étend le comportement du cas d'utilisation B (destination).

(d) Notre cas d'utilisation

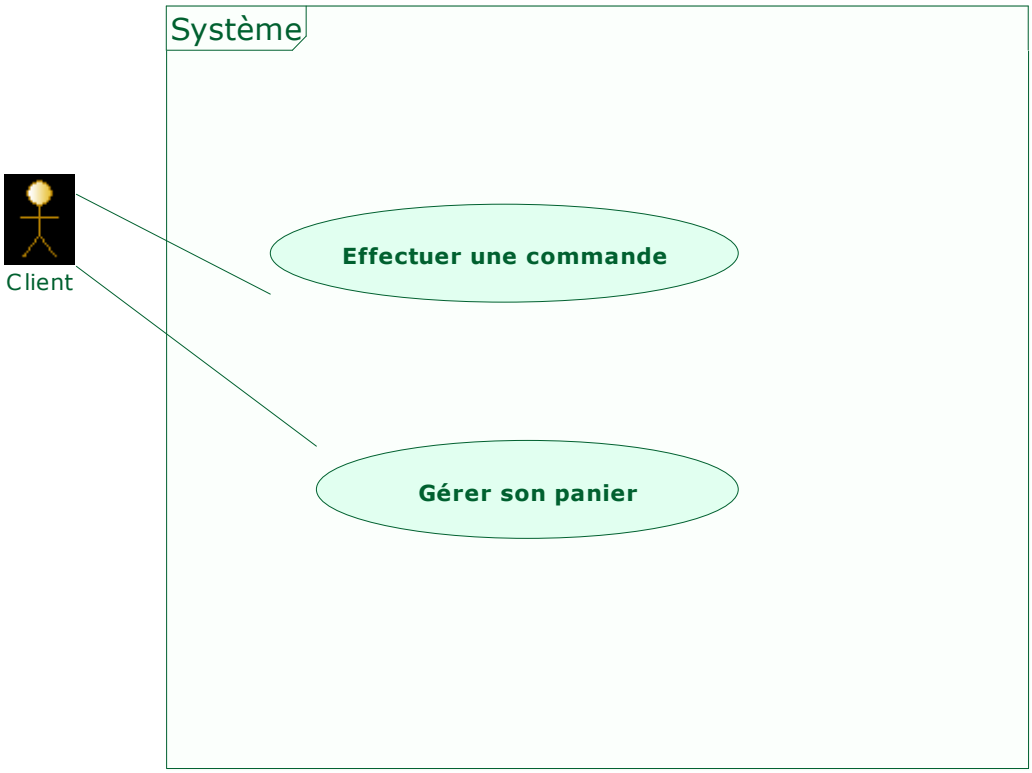


Fig. 16

Section 5.04 Diagramme de Séquence Système

(a) Réalisation d'un diagramme de séquence système

Le diagramme de séquence système montre les interactions entre les objets d'un point de vue temporel. Il insiste sur la chronologie des envois de messages.

Pour utiliser les outils (les icônes), il suffit de cliquer sur l'icône et ensuite de cliquer dans l'éditeur graphique.

Pour personnaliser le texte de chaque élément, il suffit de « double-cliquer » sur celui-ci.

(b) Eléments que l'on peut créer dans un diagramme de séquence système

Icône	Description
	Pointeur flèche
	Créer une ligne de vie
	Créer un usage d'interaction
	Créer une décomposition de part
	Créer une gate
	Créer un invariant d'état
	Créer un fragment composé
	Créer une opérande d'interaction
	Créer une spécification d'exécution
	Créer un message asynchrone
	Créer un message synchrone
	Créer un message de réponse
	Créer un message de création
	Créer un message de destruction
	Créer une note
	Créer un flux d'information

(c) Explications sur la construction d'un diagramme de séquence système

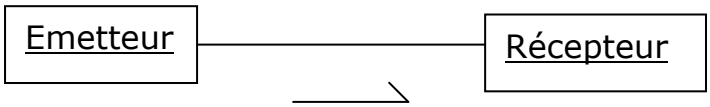
Pour élaborer un diagramme de séquence, on commence par créer les objets et ensuite les messages entre ces derniers.

Plusieurs catégories de messages :

- Les constructeurs : ils créent les objets.
- Les destructeurs : ils détruisent les objets.
- Les sélecteurs : ils renvoient toute une partie de l'état d'un objet (ex : déclencher une opération).

Un message synchrone : l'émetteur est bloqué tant que le récepteur n'a pas traité le message.

Un message asynchrone n'interrompt pas l'exécution de l'émetteur. Ce dernier envoie le message sans savoir si le message sera traité par le destinataire.



(d) Notre diagramme de séquence système « Gérer son panier »

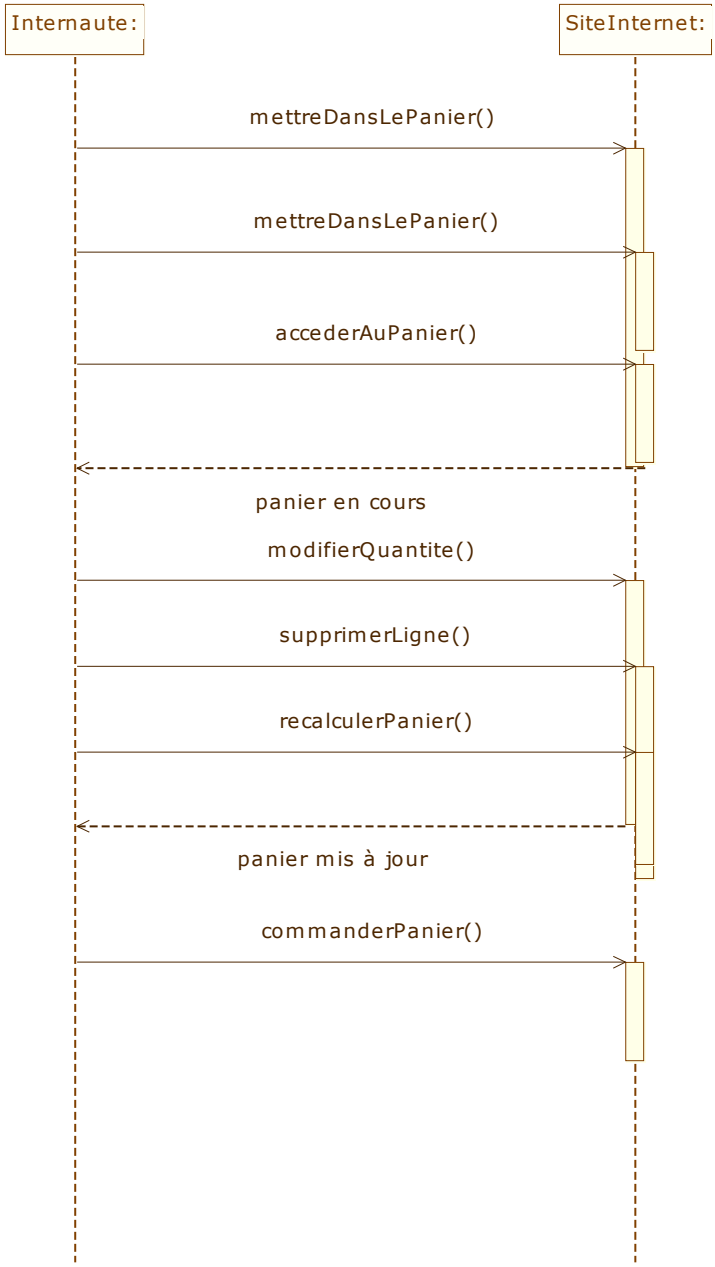


Fig. 17

(e) Notre diagramme de séquence système « Effectuer une commande »

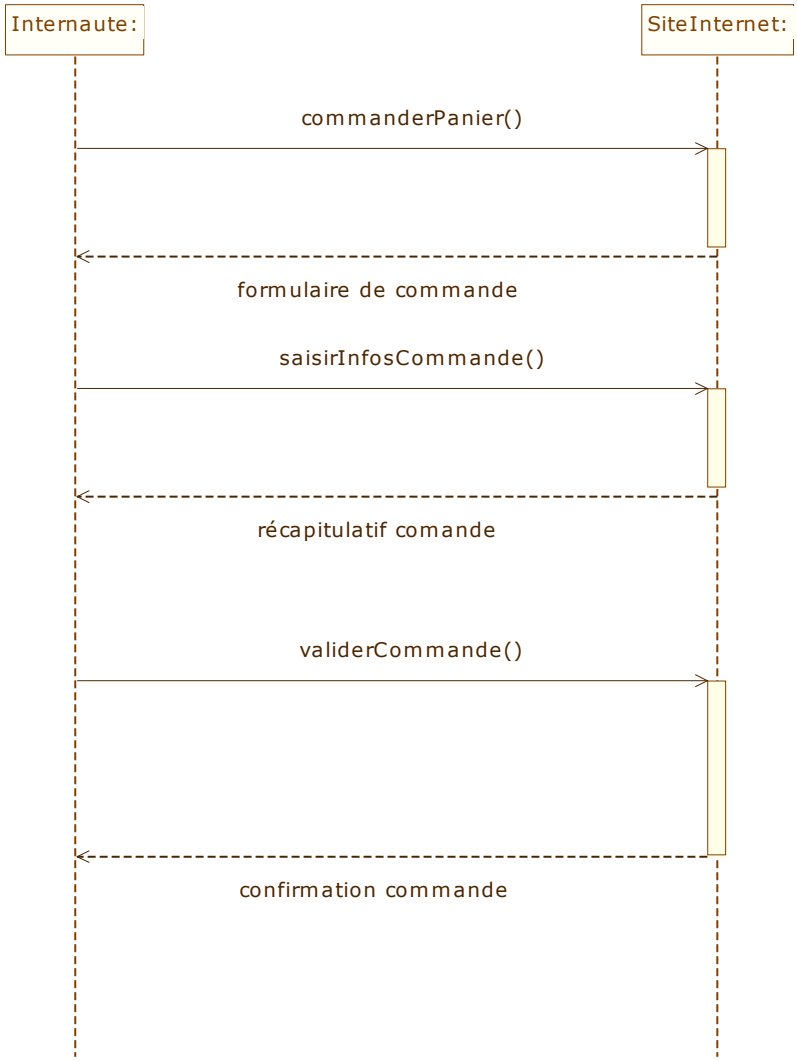


Fig. 18

Section 5.05 Le module Praxeme

Le module Praxeme permet de modéliser selon les différents aspects de Praxeme, en structurant ceux-ci dans des modèles séparés. Ce module supporte ainsi la modélisation de la sémantique métier, de l'organisation et des processus métier (aspect pragmatique Praxeme) , des architectures logiques des systèmes, des architectures matérielles et du logiciel en offrant une continuité depuis l'architecture d'entreprise jusqu'aux modèles logiciels UML. Il guide méthodologiquement en structurant les modèles selon les points de vues, les types de problèmes rencontrés et les préoccupations existant sur un système, et les interlocuteurs concernés.

Le module Praxeme est une extension de Objecteering. Il se présente sous la forme d'un fichier «.mdac ».

Il est téléchargeable à l'adresse suivante :

http://www.objecteering.fr/praxeme.php

Chapitre 6. Cohérence

La cohérence est l'un des maîtres mots de cet outil.

Il s'agit d'une cohérence de tous les instants qui se concrétise aussi bien dans les diagrammes que dans le code.

En fait, un contrôle est opéré afin de ne proposer que des options valides et d'alerter si on tente de réaliser des opérations invalides.

De plus, par cohérence il faut comprendre que toute opération effectuée sur le modèle est immédiatement répercutée dans l'arborescence du projet et ainsi par effet de bord dans les autres modèles.

Chapitre 7. Composants MDA

Section 7.01 Présentation

De nombreux composants sont disponibles dès l'installation du logiciel comme ceux que nous avons testé, à savoir la génération de documentation et de code Java.

Mais tout l'intérêt d'un tel produit est de permettre de créer ses propres composants MDA.

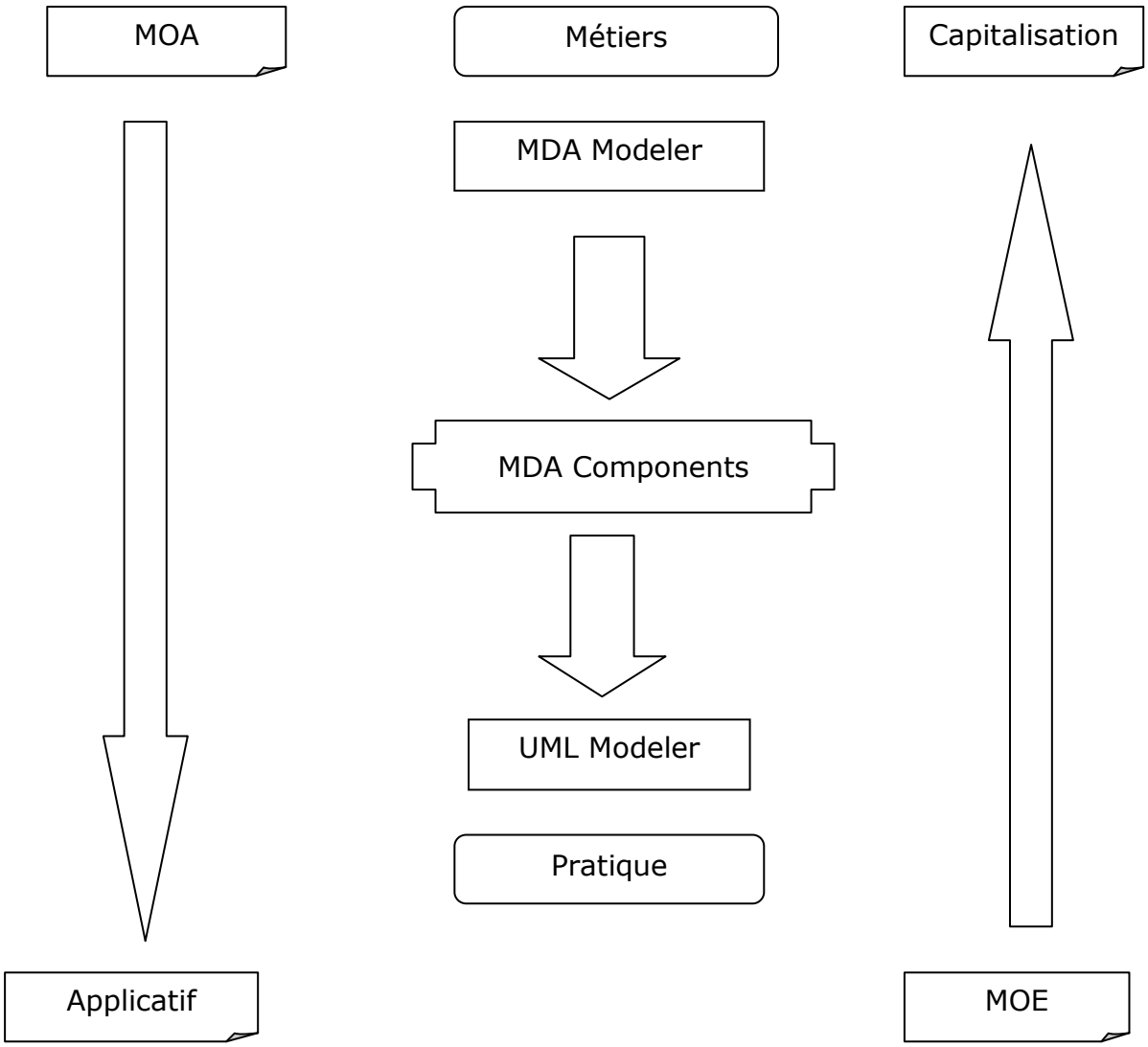
Ceci est faisable grâce au MDA Modeler qui permet aux utilisateurs de définir des opérations de production sur les modèles.

Ainsi les métiers peuvent capitaliser leurs savoir-faire et les institutionnaliser afin que ceux-ci puissent être facilement utilisés et rapidement exploités.

Cette construction même de composants MDA s'effectue par modélisation limitant ainsi au minimum les travaux de programmation.

Ces composants sont ensuite exploités par UML Modeler d'une façon pragmatique.

Ci-dessous une schématisation de l'interaction MDA Modeler vs UML Modeler :


MOA
Métiers
Capitalisation
MDA Modeler
MDA Components
UML Modeler
Pratique
Applicatif
MOE

Section 7.02 Documentation

Nous avons mis en œuvre un composant MDA de type générateur pour établir une documentation à partir du diagramme de classe de notre apllication.

(a) Démonstration

- Déployer un composant MDA et générer la documentation
- Documentation

Section 7.03 Génération de code

Bien qu'elle puisse se faire dans les langages suivants, C#, C++, Java, SQL, Corba, Fortran, etc., nous avons opté pour le langage Java, en partie pour sa portabilité et son adaptation naturelle aux problématiques du web.

Cependant, l'implémentation du corps des méthodes dans Objecteering même est extrêmement fastidieuse.

Heureusement, un mécanisme de tag permet, grâce aux @objingid exploitables sous Eclipse et Visual Studio, une fois le code généré de l'exploiter sous un IDE offrant toutes les capacités actuelles des outils dédiés au développement.

Ces tags permettent un retour aux modèles Objecteering en conservant l'apport extérieur.

(a) Démonstrations

- Insérer du code
- Générer le code

Exemple de JavaCode :

```
import com.softeam.objecteering.javadeveloper.annotations.objingid;

@objingid ("679215112:7")

public class HistoriqueMouvement {

  /**

    * 0: suppression article

    * 1: ajout article

    */

  @objingid ("679215112:50")

  private boolean mouvementArticle;
```

```
/**
 * Get accessor for mouvementArticle
 * @return  value of mouvementArticle
 */
public boolean isMouvementArticle ()  {
    return this.mouvementArticle;
}
```

(b) Hyper-généricité en J

J est un langage objet avec une syntaxe de type Java.

L'intérêt est que les corps des méthodes est développé une seule fois et peut ensuite être utilisé grâce à différents composants MDA pour générer plusieurs codes.

J est lancé au niveau du méta-model et peut interagir avec le modèle.

Cependant une critique peut être émise à ce sujet, car cela implique d'apprendre un nouveau langage, tout en considérant que ce type d'approche ne permet pas forcément d'exploiter toutes les subtilités d'un langage.

Par ailleurs, tout langage interprété comme c'est le cas de J ou même des langages tournants sur des machines virtuelle comme Java, sont déjà affranchie du support.

L'avantage indéniable est l'interaction avec le model.

La programmation J implique l'utilisation du composant « MDA Modeler ».

Exemple de JCode :

```
Package:listClasses () Method definition
{
    OwnedElementClass  // Going through the Package
                          classes
   {
      StdOut.write(NL, "CLASS:", Name);
      PartOperation.<select (Visibility == Public)
      // Going through the method's public classes
      {
         StdOut.write(NL, " Public method:", Name);
      }
   }
}           // End of "listClasses
```


Chapitre 8. Conclusion

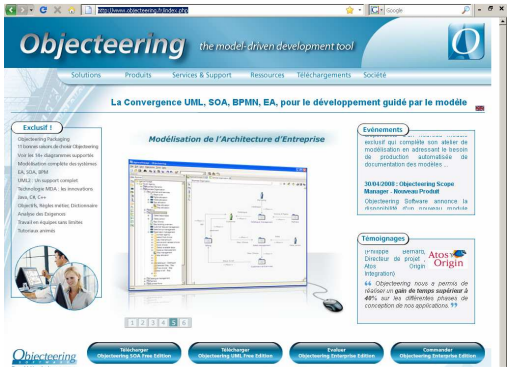
Bien que très puissant, nous n'avons pas pu appréhender réellement ce formidable outil de modélisation par manque de recul sur les métiers qu'il implique.

Par ailleurs, son orientation clairement MOA conduisant vers la MOE en fait un outil plus difficile à aborder en l'état actuel de nos connaissances, qu'un outil de type NetBeans clairement plus orienté développement qui intègre aussi une démarche de modélisation.

Dans ce contexte et devant l'envergure des possibilités offertes, nous n'avons pas été capables de discerner l'utile du nécessaire pour développer une petite application, mais nous avons exploré différents aspects de ce logiciel que nous n'aurions pas appréhendé si nous nous étions contentés de rester concentré sur la production de code.

Chapitre 9. Références

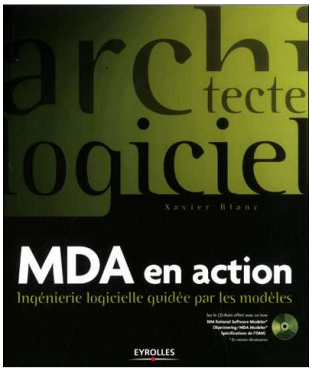
Section 9.01 Site Internet



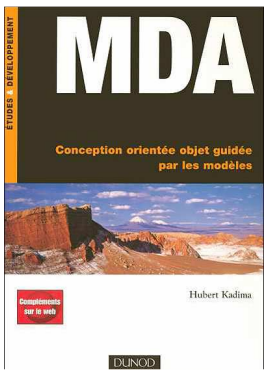
http://www.objecteering.fr/index.php

- Praxeme :
  - http://www.praxeme.org

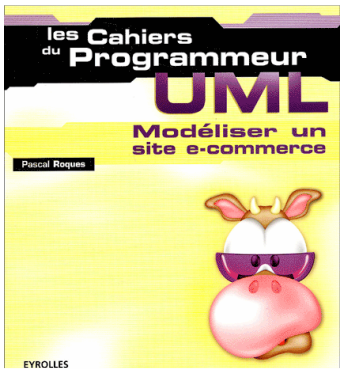
Section 9.02 Livre



Auteur	X. Blanc
Editeur	Eyrolles
Date de parution	avril 2005
Collection	Architecte Logiciel
ISBN	2212115393
Illustration	Pas d'illustrations



Auteur	Hubert Kadima
Editeur	Dunod
Date de parution	mars 2005
Collection	Infopro
Format	18 cm x 25 cm
ISBN	2100073567



Auteur	Pascal Roques
Editeur	Eyrolles
Date de parution	septembre 2002
Collection	Cahiers Du Programmeur
Format	21 cm x 24 cm
ISBN	2212110707