WP1-T12 : EVALUATION XML/XMI

RAPPORT N° CS/311-1/AJ000212/RAP/02/04/05 Version 1.0

(Version provisoire du 10/7 qui sera complétée par Charles MODIGUY)

Préparé par

CS SI

Division Opérationnelle Ingénierie Scientifique

Centre Opérationnel France Nord

16 avenue Galilée

92350 LE PLESSIS ROBINSON

Rédigé dans le cadre du projet

ACOTRIS

	NOM(S)	DATE(S)	VISA(S)
REDACTEUR(S)	Michel NAKHLE	10/07/02	
VERIFICATEUR(S)			
APPROBATEUR			

© Ce document est la propriété exclusive de **CS SI**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

FICHE DE SUIVI DES MODIFICATIONS

Version/Révision		Références		Description des modifications	Auteur(s)
Indice	Date	Page	N° §		
1.0 *2.0*	19/04/02 – 10/07/02			Première partie : l'existant. Les résultats de l'évaluation étant à compléter par Charles Modiguy *Rapport complet : Reste à produire*	Michel NAKHLE *Michel NAKHLE & Charles MODIGUY*

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

LISTE DE DIFFUSION

DIFFUSION EXTERNE

Partenaires du projet ACOTRIS	**Via site Web Acotris**

DIFFUSION INTERNE

Charles MODIGUY	**SCSA**
Nhan TRUONG TRUNG	**SCSA**
Archivage DO	**SCSA**

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

DOCUMENTS APPLICABLES ET DE REFERENCE

DOCUMENTS APPLICABLES

DOCUMENTS DE REFERENCE

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

SOMMAIRE

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

1. INTRODUCTION

1.1 CONTEXTE DE L'ETUDE

Cette étude entre dans le cadre du projet ACOTRIS.

Elle constitue la synthèse de l'étude préalable à la réalisation des passerelles XMI dans le cadre du projet ACOTRIS, passerelles dont la faisabilité doit être démontrée.

Elle retrace le travail effectué dans le sous-projet WP1 – Tâche T12 «Evaluation XML/XMI», dans sa globalité (état de l'art et outils existants), puis présente plus spécifiquement deux logiciels existants (IBM Alphaworks et Scriptor) afin d'évaluer le travail nécessaire à leur adaptation.

Elle est centrée sur le formalisme de spécification UML et le langage de description de données XMI.

L'objectif est d'étudier UML, XML et XMI afin de déterminer les règles assurant la conversion d'un ensemble d'informations modélisées en UML vers le format d'échange XMI, ainsi que la conversion réciproque d'informations XMI en UML.

1.2 ORIGINE DES INFORMATIONS

La plupart des informations contenues dans ce document sont extraites d'articles, documents et sites de l'Internet. Les principaux sites sont :

Descriptif	URL
Le site de IBM Alphaworks	http://www.alphaworks.ibm.com
Le site de SodiFrance	http://www.sodifrance.fr
Le site de l'OMG	http://www.omg.org
Le site du W3C	http://www.w3c.org
Le site de XMLsoftware	http://www.xmlsoftware.com

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

2. GLOSSAIRE

Acronyme	Nom	Définition
ASN1	Abstract Syntax Notation One	Grammaire ou métalangage spécifique de description de messages tourné principalement vers l'écriture de protocoles de communication ; normes ISO 8824/ITU X.208 et ISO 8825/ITU X.209
API	Application Programming Interface	Interface de programmation applicatifs
Biztalk	The BizTalk Framework	Ensemble de spécifications pour la conception et le développement de systèmes de messagerie basés sur XML
CDATA	Character Data	Elément de la syntaxe XML permettant d'insérer des chaînes de caractères
CML	Chemical Markup Language	Langage de description pour la représentation XML de molécules chimiques
CSS (1,2 et 3)	Cascading Style Sheet	Langage de description de feuilles de style
CORBA	Common Object Request Broker Architecture	Concept permettant à deux applications de communiquer entre elles sans se soucier de leur localisation ou de leur mode de conception
DCD	Document Content Description	Langage de description de modèle XML
DDML	Document Description Markup Language	Langage de description de modèle XML
DOM (1 et 2)	Document Object Model	Mécanisme permettant de manipuler des contenus HTML et XML "parsed"
DTD	Document Type Description	Langage de description de modèle XML
EDI **IDE**	Environnement de développement Intégré Integrated Development Environment	

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

FTP	File Transfer Protocol	Protocole
GUI	Graphic User Interface	Interface graphique
HTML	HyperText Markup Language	Métalangage
HTTP	HyperText Transfer Protocol	Protocole
HyTime	HyTime	Norme complémentaire à SGML pour la localisation (lien hypertexte) d'objets d'information
JDBC	Java Data Base Connectivity	API Java pour l'exécution de requêtes SQL.
LDAP	Lightweight Directory Access Protocol	Protocole
MFC	Microsoft Foundation Classes	
MOF	Meta Object Facility	Modèles de représentation de l'interopérabilité des connaissances
Namespaces	Namespaces (espace nominal)	Mécanisme de pré-fixage des balises pour différencier 2 noms identiques de 2 DTD différentes
ODBC	Open Data Base Connectivity	API standard de connexion à toute base de données possédant un pilote ODBC et permettant l'exécution de requêtes SQL
OMG	Object Management Group	Groupe de travail
OSI	Open System Interconnection	Modèle pour le développement d'applications
PHP		Langage basé sur Java, HTML et PERL pour la création rapide de pages Internet générées dynamiquement
PointeursTEI	Text Encoding Initiative	Mécanisme conçu pour SGML et destiné au codage de textes littéraires de tout type sur lequel se basent les pointeurs XML
RDF	Resource Description Framework	Base pour le traitement de méta-données
RTF	Rich-Text Format	Format Texte créé par Microsoft
SAX	Simple API to XML	

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

SGML	Standard Generalized Markup Language	Métalangage à partir duquel est élaboré XML
SMIL	Synchronized Multimedia Integration Language	Langage de présentation de documents XML
sNets	Représentation basée sur les réseaux sémantiques	Formalisme
SOX	Specification Oriented-Object eXchange	Langage de description de modèle XML
UML	Unified Modeling Language	Langage de spécification et de conception d'application
URI	Unique Resource Identifier	Identifiant unique d'une ressource. Par exemple, les URL sont des URI particuliers
UXF	UML eXchange Format	Format XML permettant des échanges de données entre référentiels UML
UXF	UML eXchange Format	Format XML permettant des échanges de données entre référentiels UML
WYSIWYG	«What you see is what you get»	Concept
XDR	eXternal Data Representation Standard	Standard de description et d'encodage de données utilisé pour l'échange d'informations
Xlink	eXtensible Linking Language	Langage de description de liens
XMI	XML Metadata Interchange	Format permettant des échanges de méta-données entre référentiels
XML	eXtensible Markup Language	Métalangage d'échanges de données
XML Schema	XML Schema	Langage de description de la structure et du contenu (contraintes) de documents XML
XML-DATA		Langage de description de modèle XML orienté base de données
XML-QL / QL	XML Query Language for XML	
Xpath	Xpath Language	Langage de localisation des nœuds d'un document XML
Xpointer	XML Pointer Language	Langage de description du positionnement des liens

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

XSD	XML Schema Description	Syntaxe XML Schéma
XSL	eXtensible Stylesheet Language	Langage de description de feuilles de style
XSLT	XSL Transformations	Langage de transformation de documents XML

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

3. INTRODUCTION AUX FORMALISMES XML ET XMI

3.1 LE FORMALISME XML

Le formalisme XML (Extended Markup Language) permet la représentation structurée d'informations dans un format texte. Il peut par conséquent être utilisé comme format syntaxique de transport de modèles et de méta-modèles. La structure de l'information est alors définie dans un fichier annexe au format DTD (Document Type Description). L'avantage d'un tel formalisme est de permettre l'échange d'informations dès lors que l'on base celle-ci sur une DTD normalisée.

Voici un exemple de document XML représentant un livre et sa table des matières :

```
<?xml version="1.0" ?>
<!DOCTYPE Livre "Livre.dtd">
<Livre Auteur="Michel Nakhlé et Charles Modiguy">
  <Titre>Rapport</ Titre >
  <Chapitre id="1">
    Premier Chapitre. Introduction.
  </Chapitre >
  <Chapitre id="2">
    Second Chapitre. Glossaire.
  </Chapitre >
</Livre>
```

Dans ce formalisme, nous trouvons la notion d'élément. Les éléments sont définis avec un marqueur de début et un marqueur de fin. Le marqueur de début constitué simplement du nom de l'élément tandis que le marqueur de fin est constitué du nom de l'élément préfixé du caractère "/".

Dans l'exemple précédent, nous avons les éléments

- Rapport représentant les rapports,
- Titre représentant le titre des rapports et
- Chapitre représentant l'intitulé d'un chapitre.

Ces éléments peuvent ensuite disposer d'attributs. Les attributs correspondent généralement au niveau le plus fin de décomposition des éléments. Dans l'exemple précédent, nous avons l'attribut Auteur défini sur l'élément Rapport et représentant le nom de l'auteur du rapport et l'attribut id sur l'élément Chapitre indiquant le numéro du chapitre.

Le rôle du fichier au format DTD est de décrire tous les éléments que l'on est susceptible de rencontrer dans le fichier XML ainsi que tous les attributs que ces éléments sont susceptibles de disposer.

Ainsi, la DTD nommée livre.dtd qui a permis de construire le fichier XML précédent est la suivante :

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

```
<!DOCTYPE Livre [
    <!ELEMENT Rapport (Titre, Chapitre+)>
    <!ATTLIST Rapport Auteur CDATA #REQUIRED>
    <!ELEMENT Titre (#PCDATA)>
    <!ELEMENT Chapitre (#PCDATA)>
    <!ATTLIST Chapitre id ID #REQUIRED>
]>
```

Cette DTD décrit les trois éléments (Rapport, Titre et Chapitre) ainsi que leurs attributs.

Actuellement, l'utilisation de DTD pour décrire le contenu des fichiers XML est en train de disparaître au profit des schémas. Les schémas sont des fichiers au format XML (ce qui n'était pas le cas des fichiers DTD) et proposent un certain nombre de facilités pour la définition de la structure des fichiers XML.

L'un des avantages de XML est sa souplesse et sa lisibilité tandis que ses désavantages sont l'aspect "verbeux" du langage entraînant rapidement des fichiers d'une taille importante et le fait que les données représentées dans un fichier XML le sont sous la forme d'un arbre. En effet, dès que l'on souhaite représenter des graphes dans un document XML, il est nécessaire de définir des références qui vont alors à l'encontre de la souplesse et la lisibilité. Or les modèles et les méta-modèles sont plus généralement constitués de graphes que d'arbres.

Soit le cas de la représentation des programmes présents sur un site de la Figure 1.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

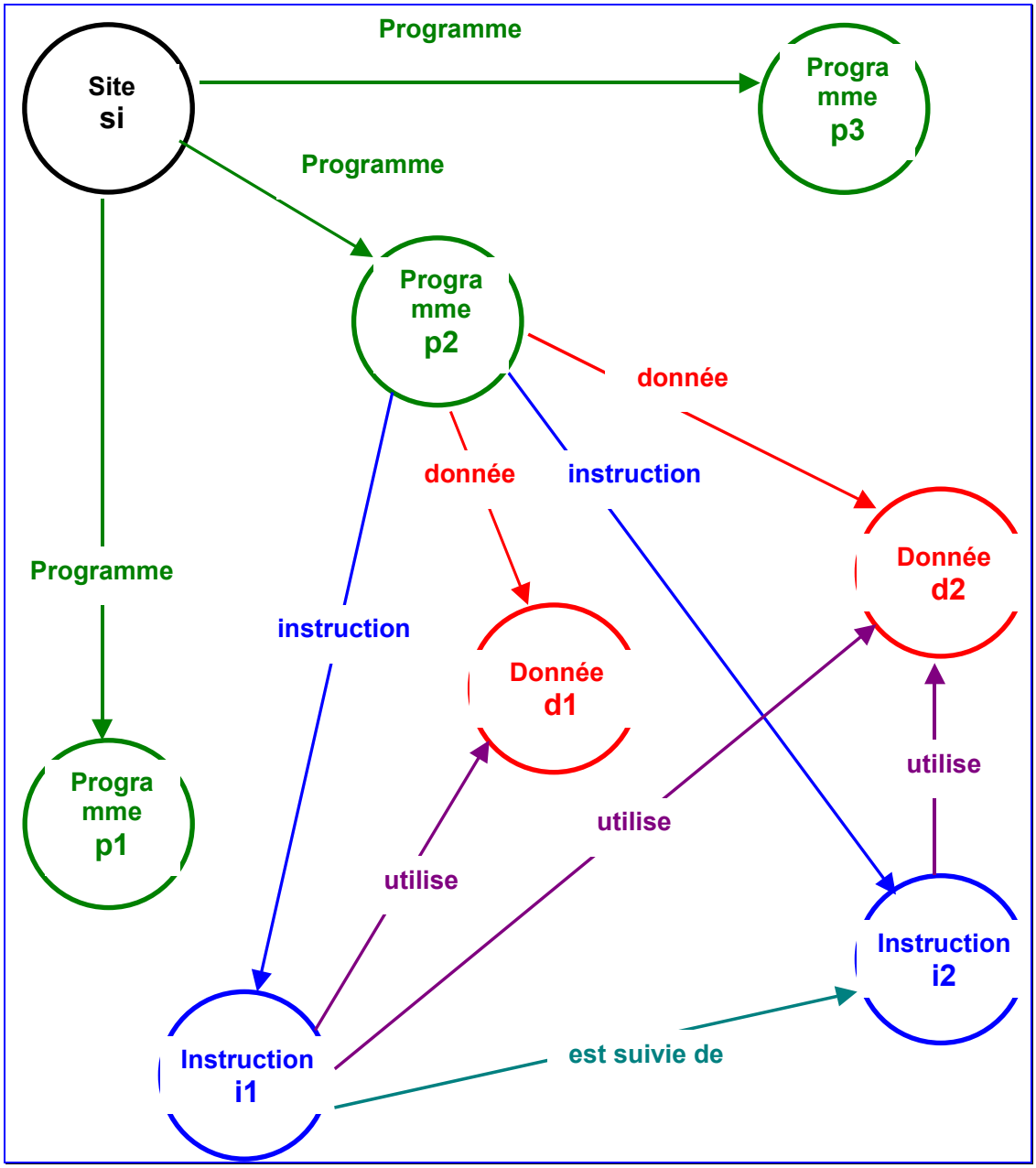


Figure 1 : Description des programmes d'un site, représentée sous la forme d'un réseau sémantique

Le méta-modèle de ces informations pourra être exprimé à l'aide d'une DTD de la façon suivante :

```
<!DOCTYPE Site [
    <!ELEMENT Site ( Programme+ )>
    <!ATTLIST Site nom CDATA #REQUIRED>
    <!ELEMENT Programme ( Donnee+, Instruction+)>
```


© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

```
    <!ATTLIST Programme nom CDATA #REQUIRED>
    <!ELEMENT Donnee EMPTY>
    <!ATTLIST Donnee id ID #REQUIRED, nom CDATA #Required >
    <!ELEMENT Instruction ( DonneeUtilisee+)>
    <!ATTLIST Instruction id ID #REQUIRED,
            nom CDATA #Required,
            instructionSuivante CDATA #IMPLIED>
    <!ELEMENT DonneeUtilisee EMPTY>
    <!ATTLIST DonneeUtilisee id CDATA #REQUIRED>
]>
```

On remarque ici que le lien d'utilisation entre une instruction et une donnée nécessite un nouveau type d'élément appelé DonneeUtilisee. Ceci est du au fait que les modèles représentés sont des graphes et que ce formalisme ne représente que des données arborescentes. Il est donc nécessaire de passer par des identifiants et des références à ces identifiants pour représenter les modèles.

Ainsi les informations de la Figure 1 pourront être représentées en utilisant ce formalisme de la façon suivante :

```
<?xml version="1.0" ?>
<!DOCTYPE Site "Site.dtd">
<Site nom="s">
  <Programme id="#1" nom="p1">
  </Programme>
  <Programme id="#2" nom="p2">
    <Donnee id="#4" nom="d1" >
    </Donnee>
    <Donnee id="#5" nom="d2" >
    </Donnee>
    <Instruction id="#6" nom="i1" instructionSuivante="#6">
      <DonneeUtilisee id="#4">
      </DonneeUtilisee>
    </Instruction>
    <Instruction id="#7" nom="i2">
      <DonneeUtilisee id="#4">
      </DonneeUtilisee>
      <DonneeUtilisee id="#5">
      </DonneeUtilisee>
    </Instruction>
  </Programme>
```


© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

```
    <Programme id="#3" nom="p3">
    </Programme>
  </Site>
```

La représentation de ces informations devient alors rapidement complexe et la manipulation de celles-ci se complexifie également.

3.2 LE FORMAT D'INTERCHANGE DE META DONNEES, XMI

3.2.1 Le formalisme XMI

XMI (XML Metadata Interchange) est un format d'échange de modèles basé sur la sémantique du MOF et sur la syntaxe de XML. Ce format est issu d'une mise en correspondance des descriptions MOF de méta-modèles et de la façon de définir celles-ci dans une DTD.

L'application de cette mise en correspondance sur un méta-modèle permet d'obtenir une DTD standard pour l'échange des modèles décrits à l'aide de ce méta-modèle. Il existe ainsi une DTD standard permettant l'échange de modèles UML (obtenue en appliquant ces règles de mise en correspondance au méta-modèle de UML exprimé à l'aide du MOF) ainsi qu'une DTD standard permettant l'échange de méta-modèles MOF (obtenue en appliquant ces règles de mise en correspondance au méta-modèle du MOF lui-même).

3.2.2 Objectif de XMI

L'objet principal de XMI est de permettre l'échange de méta données entre outils de modélisation basés sur UML et la communication des répertoires de méta données basés sur le MOF deux standards de l'OMG.

L'effet économique et technique est important, parce que jusqu'ici il était à peu près impossible à une communauté de travail, un groupe d'entreprises… de travailler sur des modèles communs en utilisant des outils de conception provenant de fournisseurs différents.

XMI sert d'outil permettant de faire travailler de façon concourante de tels produits. L'annonce d'une offre commune de Rational et de IBM, autour de leurs outils de génie logiciel, utilisant XMI pour les échanges, en est une illustration.

XMI se fonde sur les trois standards XML, UML et MOF.

- UML est un formalisme orienté objet de conception et de documentation d'applications,
- MOF est une technologie de définition et de représentation de méta données en tant qu'objets CORBA. Pour mémoire, CORBA est un concept permettant à deux applications de communiquer entre elles sans se soucier de leur localisation ou de leur mode de conception.

Les spécifications de XMI consistent à proposer un ensemble de règles de transformation de structures MOF en DTD XML, des DTD pour UML et MOF, un ensemble de règles de production de documents XML pour manipuler des méta données MOF.

XMI est donc la rencontre entre deux mondes importants, celui des objets et celui des structures de données et documents.

XMI est développé par l'OMG. L'Object Management Group rassemble tous les acteurs industriels concernés par les technologies objet, c'est-à-dire pratiquement tous les industriels informatique télécom et de nombreux grands utilisateurs. La démarche objet suppose une

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

modélisation poussée des données et des processus, pour identifier les classes d'objets, les types d'instances, les actions possibles, les dialogues et transactions entre objets. Le but de l'OMG est la communication entre toutes les applications, voire en fait la disparition à terme du concept même d'application au profit des relations entre des objets. Ceux ci sont avant tout définis par des interfaces, qui sont en fait ce que le monde extérieur perçoit. La démarche de l'OMG, une fois défini un langage d'interface (IDL,) a été de permettre la mise en relations de systèmes divers, quelle que soit leur nature – et donc des systèmes existants – en créant des vues de ces systèmes capables de communiquer. Entre ces systèmes, les Object Request Brokers et autres logiciels de middleware établissent alors la relation.

Deux éléments importants ont donc été produits :

1. L'un est le MOF, qui permet de décrire les structures et caractéristiques des systèmes en présence, ce qui permet au middleware de définir les objets communicants.
2. L'autre est le langage de modélisation, UML qui est un langage graphique conçu pour décrire, spécifier, développer et documenter un système informatique. UML est le résultat de l'unification de méthodes orientées objet, avec pour objectifs la modélisation, de la conception à la mise en œuvre, le développement d'un formalisme de représentation de la phase de modélisation. La version UML 1.0, a fait l'objet d'une standardisation en 1997 par l'OMG.

Le succès de UML a été très rapide. En effet, jusqu'ici, la modélisation des systèmes produisait des descriptions et des schémas à partir de logiciels divers mais ces descriptions ne pouvaient être transférées autrement que sous forme de textes et de dessins. Le besoin d'un langage commun devenait pressant.

XMI et les règles d'utilisation correspondantes vont, dans ce contexte, jouer plusieurs rôles :

- permettre le traitement et la manipulation des méta données (création de DTD exprimant une structure, génération et manipulation de documents XML correspondants)
- permettre la mise au point des modèles pour des ensembles d'acteurs hétérogènes en réalisant l'échange des notations UML
- fournir un ensemble de règles pour traduire les modèles UML et produire les messages correspondant à des scénarios d'échange. Cette dernière possibilité est désormais intégrée dans les démarches de groupes EDI.

Le schéma suivant esquisse le dialogue entre deux entités A et B qui souhaitent développer des échanges d'information. Ces deux entités peuvent être des éléments d'une même entreprise comme des partenaires de commerce, ou encore deux applications sur deux domaines différents. Elles ont des vues du monde qui s'expriment dans des modèles, et leurs relations représentent obligatoirement une vue commune pour une partie de leurs activités. Une fois exprimé le scénario d'échange ou d'interaction en UML, l'expression en XMI des éléments relatifs à leur relation permet la mise au point progressive d'un scénario commun.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

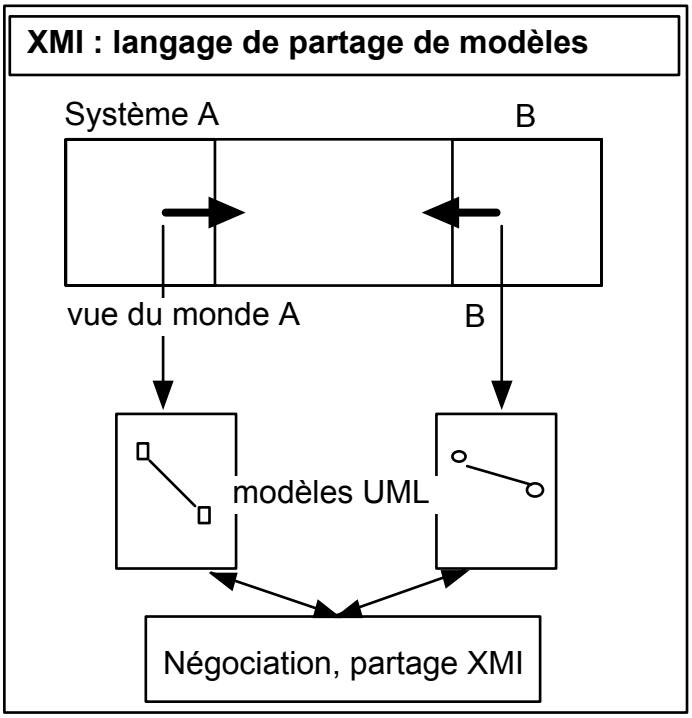


Figure 2 : Schéma du dialogue entre deux entités A et B qui souhaitent développer des échanges d'information

3.2.3 Exemple de fichier XMI généré à partir d'une modélisation UML par l'éditeur Rose2000 de Rational

L'exemple choisi est un exemple extrêmement simple car la taille des fichiers est beaucoup trop importante pour des modélisations plus importantes (à partir de dix classes, les fichiers représentent plus d'une dizaine de pages).

Modélisation UML :

DTD interne aux fichiers XMI de Rose2000.

```
<?xml version = '1.0' encoding = 'ISO-8859-1' ?>
<!DOCTYPE XMI SYSTEM 'uml.dtd' [
<!ELEMENT uisToolName (#PCDATA)>
<!ELEMENT uisDiagramName (#PCDATA)>
<!ELEMENT uisDiagramStyle (#PCDATA)>
<!ELEMENT uisDiagramPresentation (Foundation.Auxiliary_Elements.Presentation*)>
<!ELEMENT UISDiagram (uisDiagramName, uisToolName, uisDiagramStyle,
                                                uisDiagramPresentation*)>
<!ATTLIST UISDiagram xmi.id ID #REQUIRED>
```


© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

```
                <!ELEMENT uisOwnedDiagram (UISDiagram*)>
                <!ELEMENT UISModelElement (uisOwnedDiagram*)>
                <!ATTLIST UISModelElement xmi.id ID #REQUIRED>
                <!ENTITY bs "&#38;#08;">
                <!ENTITY vt "&#38;#11;">
                <!ENTITY ff "&#38;#12;">
                ]>
<XMI xmi.version = '1.0'>
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>Unisys.IntegratePlus.2</XMI.exporter>
      <XMI.exporterVersion>4.0.1</XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name = 'UML' xmi.version = '1.1'/>
  </XMI.header>
  <XMI.content>
    <Model_Management.Model xmi.id = 'G.1'>
      <Foundation.Core.ModelElement.name>Voiture</Foundation.Core.ModelElement.name>
      <Foundation.Core.ModelElement.visibility xmi.value = 'private'/>
      <Foundation.Core.GeneralizableElement.isRoot xmi.value = 'false'/>
      <Foundation.Core.GeneralizableElement.isLeaf xmi.value = 'false'/>
      <Foundation.Core.GeneralizableElement.isAbstract xmi.value = 'false'/>
      <XMI.extension xmi.extender = 'Unisys.IntegratePlus.2'>
        <XMI.reference xmi.idref = 'G.5'/>
      </XMI.extension>
      <Foundation.Core.Namespace.ownedElement>
        <Foundation.Core.Class xmi.id = 'S.10001'>
          <Foundation.Core.ModelElement.name>VOITURE</Foundation.Core.ModelElement.name>
          <Foundation.Core.ModelElement.visibility xmi.value = 'public'/>
          <Foundation.Core.GeneralizableElement.isRoot xmi.value = 'true'/>
          <Foundation.Core.GeneralizableElement.isLeaf xmi.value = 'true'/>
          <Foundation.Core.GeneralizableElement.isAbstract xmi.value = 'false'/>
          <Foundation.Core.Class.isActive xmi.value = 'false'/>
          <Foundation.Core.ModelElement.namespace>
            <Model_Management.Model xmi.idref = 'G.1'/> <!-- Voiture -->
          </Foundation.Core.ModelElement.namespace>
          <Foundation.Core.Classifier.associationEnd>
            <Foundation.Core.AssociationEnd xmi.idref = 'G.3'/> <!-- {395374F302A6} -->
          </Foundation.Core.Classifier.associationEnd>
          <Foundation.Core.ModelElement.taggedValue>
            <Foundation.Extension_Mechanisms.TaggedValue>

<Foundation.Extension_Mechanisms.TaggedValue.tag>persistence</Foundation.Extension_Mechanisms.
                                                              TaggedValue.tag>

<Foundation.Extension_Mechanisms.TaggedValue.value>transient</Foundation.Extension_Mechanisms.
```

Add-In de Rose 2000 développé par Unisys et permettant l'export et l'import XMI.

Définition du modèle UML.

Définition de la classe VOITURE

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

```
                                                                    TaggedValue.value>
            </Foundation.Extension_Mechanisms.TaggedValue>
          </Foundation.Core.ModelElement.taggedValue>
        </Foundation.Core.Class>
        <Foundation.Core.Class xmi.id = 'S.10002'>
          <Foundation.Core.ModelElement.name>PIECES</Foundation.Core.ModelElement.name>
          <Foundation.Core.ModelElement.visibility xmi.value = 'public'/>
          <Foundation.Core.GeneralizableElement.isRoot xmi.value = 'true'/>
          <Foundation.Core.GeneralizableElement.isLeaf xmi.value = 'true'/>
          <Foundation.Core.GeneralizableElement.isAbstract xmi.value = 'false'/>
          <Foundation.Core.Class.isActive xmi.value = 'false'/>
          <Foundation.Core.ModelElement.namespace>
            <Model_Management.Model xmi.idref = 'G.1'/> <!-- Voiture -->
          </Foundation.Core.ModelElement.namespace>
          <Foundation.Core.Classifier.associationEnd>
            <Foundation.Core.AssociationEnd xmi.idref = 'G.4'/> <!-- {395374F302A7} -->
          </Foundation.Core.Classifier.associationEnd>
          <Foundation.Core.ModelElement.taggedValue>
            <Foundation.Extension_Mechanisms.TaggedValue>

<Foundation.Extension_Mechanisms.TaggedValue.tag>persistence</Foundation.Extension_Mechanisms.
                                                                    TaggedValue.tag>

<Foundation.Extension_Mechanisms.TaggedValue.value>transient</Foundation.Extension_Mechanisms.
                                                                    TaggedValue.value>
            </Foundation.Extension_Mechanisms.TaggedValue>
          </Foundation.Core.ModelElement.taggedValue>
        </Foundation.Core.Class>
        <Foundation.Core.Association xmi.id = 'G.2'>
          <Foundation.Core.ModelElement.name></Foundation.Core.ModelElement.name>
          <Foundation.Core.ModelElement.visibility xmi.value = 'private'/>
          <Foundation.Core.GeneralizableElement.isRoot xmi.value = 'false'/>
          <Foundation.Core.GeneralizableElement.isLeaf xmi.value = 'false'/>
          <Foundation.Core.GeneralizableElement.isAbstract xmi.value = 'false'/>
          <Foundation.Core.Association.connection>
            <Foundation.Core.AssociationEnd xmi.id = 'G.3'>
              <Foundation.Core.ModelElement.name></Foundation.Core.ModelElement.name>
              <Foundation.Core.ModelElement.visibility xmi.value = 'public'/>
              <Foundation.Core.AssociationEnd.isNavigable xmi.value = 'false'/>
              <Foundation.Core.AssociationEnd.isOrdered xmi.value = 'false'/>
              <Foundation.Core.AssociationEnd.aggregation xmi.value = 'shared'/>

<Foundation.Core.AssociationEnd.multiplicity>1..1</Foundation.Core.AssociationEnd.multiplicity>
              <Foundation.Core.AssociationEnd.changeable xmi.value = 'none'/>
              <Foundation.Core.AssociationEnd.targetScope xmi.value = 'instance'/>
              <Foundation.Core.AssociationEnd.type>
                <Foundation.Core.Class xmi.idref = 'S.10001'/> <!-- VOITURE -->
```

Définition de la classe PIECES

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

```
            </Foundation.Core.AssociationEnd.type>
          </Foundation.Core.AssociationEnd>
          <Foundation.Core.AssociationEnd xmi.id = 'G.4'>
            <Foundation.Core.ModelElement.name></Foundation.Core.ModelElement.name>
            <Foundation.Core.ModelElement.visibility xmi.value = 'public'/>
            <Foundation.Core.AssociationEnd.isNavigable xmi.value = 'false'/>
            <Foundation.Core.AssociationEnd.isOrdered xmi.value = 'false'/>
            <Foundation.Core.AssociationEnd.aggregation xmi.value = 'none'/>
<Foundation.Core.AssociationEnd.multiplicity>1..*</Foundation.Core.AssociationEnd.multiplicity>
            <Foundation.Core.AssociationEnd.changeable xmi.value = 'none'/>
            <Foundation.Core.AssociationEnd.targetScope xmi.value = 'instance'/>
            <Foundation.Core.AssociationEnd.type>
              <Foundation.Core.Class xmi.idref = 'S.10002'/> <!-- PIECES -->
            </Foundation.Core.AssociationEnd.type>
          </Foundation.Core.AssociationEnd>
        </Foundation.Core.Association.connection>
      </Foundation.Core.Association>
    </Foundation.Core.Namespace.ownedElement>
  </Model_Management.Model>  <!-- End Model G.1 -->
</XMI.content>
<XMI.extensions xmi.extender = 'Unisys.IntegratePlus.2'>
  <UISModelElement xmi.id = 'G.5'>
    <uisOwnedDiagram>
      <UISDiagram xmi.id = 'S.10005'>
        <uisDiagramName>Main</uisDiagramName>
        <uisToolName>Rational Rose 98</uisToolName>
        <uisDiagramStyle>ClassDiagram</uisDiagramStyle>
        <uisDiagramPresentation>
          <Foundation.Auxiliary_Elements.Presentation xmi.id='G.6'>
            <Foundation.Auxiliary_Elements.Presentation.geometry>
              <Foundation.Data_Types.Geometry>
                <Foundation.Data_Types.Geometry.body>
                   832, 240, 212, 126,
                </Foundation.Data_Types.Geometry.body>
              </Foundation.Data_Types.Geometry>
            </Foundation.Auxiliary_Elements.Presentation.geometry>
            <Foundation.Auxiliary_Elements.Presentation.style>
              <Foundation.Data_Types.GraphicMarker>
```

Cardinalités

Définition des caractéristiques graphiques (présentation) du modèle permettant à l'éditeur d'afficher un fichier importé correctement.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

```
        <Foundation.Data_Types.GraphicMarker.body>
            Font.Blue= 0,Font.Green= 0,Font.Red= 0,Font.FaceName=Arial,
            Font.Size= 10,Font.Bold=0,Font.Italic=0,Font.Strikethrough=0,
            Font.Underline=0, LineColor.Blue= 51,LineColor.Green= 0,
            LineColor.Red= 153,FillColor.Blue= 204,FillColor.Green= 255,
            FillColor.Red= 255, FillColor.Transparent=1, AutomaticResize=1,
            ShowAllAttributes=1,ShowAllOperations=1,ShowOperationSignature=0,
            SuppressAttributes=0,SuppressOperations=0,
        </Foundation.Data_Types.GraphicMarker.body>
      </Foundation.Data_Types.GraphicMarker>
    </Foundation.Auxiliary_Elements.Presentation.style>
    <Foundation.Auxiliary_Elements.Presentation.model>
      <Foundation.Core.Class xmi.idref = 'S.10002'/> <!-- PIECES -->
    </Foundation.Auxiliary_Elements.Presentation.model>
  </Foundation.Auxiliary_Elements.Presentation>
  <Foundation.Auxiliary_Elements.Presentation xmi.id='G.7'>
    <Foundation.Auxiliary_Elements.Presentation.geometry>
      <Foundation.Data_Types.Geometry>
        <Foundation.Data_Types.Geometry.body>
           320, 240, 222, 126,
        </Foundation.Data_Types.Geometry.body>
      </Foundation.Data_Types.Geometry>
    </Foundation.Auxiliary_Elements.Presentation.geometry>
    <Foundation.Auxiliary_Elements.Presentation.style>
      <Foundation.Data_Types.GraphicMarker>
        <Foundation.Data_Types.GraphicMarker.body>
            Font.Blue= 0,Font.Green= 0,Font.Red= 0,Font.FaceName=Arial,
            Font.Size=10,Font.Bold=0,Font.Italic=0,Font.Strikethrough=0,
            Font.Underline=0,LineColor.Blue= 51,LineColor.Green= 0,
            LineColor.Red= 153,FillColor.Blue= 204,FillColor.Green= 255,
            FillColor.Red=255,FillColor.Transparent=1,AutomaticResize=1,
            ShowAllAttributes=1,ShowAllOperations=1,ShowOperationSignature=0,
            SuppressAttributes=0,SuppressOperations=0,
        </Foundation.Data_Types.GraphicMarker.body>
      </Foundation.Data_Types.GraphicMarker>
    </Foundation.Auxiliary_Elements.Presentation.style>
    <Foundation.Auxiliary_Elements.Presentation.model>
      <Foundation.Core.Class xmi.idref = 'S.10001'/> <!-- VOITURE -->
    </Foundation.Auxiliary_Elements.Presentation.model>
  </Foundation.Auxiliary_Elements.Presentation>
  <Foundation.Auxiliary_Elements.Presentation xmi.id='G.8'>
    <Foundation.Auxiliary_Elements.Presentation.geometry>
      <Foundation.Data_Types.Geometry>
        <Foundation.Data_Types.Geometry.body>
           578, 240, 348, 54,
```


© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

```
        </Foundation.Data_Types.Geometry.body>
      </Foundation.Data_Types.Geometry>
    </Foundation.Auxiliary_Elements.Presentation.geometry>
    <Foundation.Auxiliary_Elements.Presentation.style>
      <Foundation.Data_Types.GraphicMarker>
        <Foundation.Data_Types.GraphicMarker.body>
          Association:Font.Blue= 0,Font.Green= 0, Font.Red= 0,Font.FaceName=Arial,
                    Font.Size= 10,Font.Bold=0,Font.Italic=0,Font.Strikethrough=0,
                    Font.Underline=0, LineColor.Blue= 51,
                    LineColor.Green= 0,LineColor.Red= 153,
        </Foundation.Data_Types.GraphicMarker.body>
      </Foundation.Data_Types.GraphicMarker>
    </Foundation.Auxiliary_Elements.Presentation.style>
    <Foundation.Auxiliary_Elements.Presentation.model>
      <Foundation.Core.Association xmi.idref = 'G.2'/>
                                          <!-- {VOITURE-PIECES}{395374F302A5} -->
    </Foundation.Auxiliary_Elements.Presentation.model>
  </Foundation.Auxiliary_Elements.Presentation>
  <Foundation.Auxiliary_Elements.Presentation xmi.id='G.9'>
    <Foundation.Auxiliary_Elements.Presentation.geometry>
      <Foundation.Data_Types.Geometry>
        <Foundation.Data_Types.Geometry.body>
           650, 240, 203, 54,
        </Foundation.Data_Types.Geometry.body>
      </Foundation.Data_Types.Geometry>
    </Foundation.Auxiliary_Elements.Presentation.geometry>
    <Foundation.Auxiliary_Elements.Presentation.style>
      <Foundation.Data_Types.GraphicMarker>
        <Foundation.Data_Types.GraphicMarker.body>
          Role
        </Foundation.Data_Types.GraphicMarker.body>
      </Foundation.Data_Types.GraphicMarker>
    </Foundation.Auxiliary_Elements.Presentation.style>
    <Foundation.Auxiliary_Elements.Presentation.model>
      <Foundation.Core.AssociationEnd xmi.idref = 'G.4'/> <!-- {395374F302A7} -->
    </Foundation.Auxiliary_Elements.Presentation.model>
  </Foundation.Auxiliary_Elements.Presentation>
  <Foundation.Auxiliary_Elements.Presentation xmi.id='G.10'>
    <Foundation.Auxiliary_Elements.Presentation.geometry>
      <Foundation.Data_Types.Geometry>
        <Foundation.Data_Types.Geometry.body>
           503, 240, 199, 54,
        </Foundation.Data_Types.Geometry.body>
      </Foundation.Data_Types.Geometry>
    </Foundation.Auxiliary_Elements.Presentation.geometry>
    <Foundation.Auxiliary_Elements.Presentation.style>
```


© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

```
                <Foundation.Data_Types.GraphicMarker>
                  <Foundation.Data_Types.GraphicMarker.body>
                    Role
                  </Foundation.Data_Types.GraphicMarker.body>
                </Foundation.Data_Types.GraphicMarker>
              </Foundation.Auxiliary_Elements.Presentation.style>
              <Foundation.Auxiliary_Elements.Presentation.model>
                <Foundation.Core.AssociationEnd xmi.idref = 'G.3'/> <!-- {395374F302A6} -->
              </Foundation.Auxiliary_Elements.Presentation.model>
            </Foundation.Auxiliary_Elements.Presentation>
          </uisDiagramPresentation>
        </UISDiagram>
        <UISDiagram xmi.id = 'S.10006'>
          <uisDiagramName>Main</uisDiagramName>
          <uisToolName>Rational Rose 98</uisToolName>
          <uisDiagramStyle>ModuleDiagram</uisDiagramStyle>
        </UISDiagram>
      </uisOwnedDiagram>
    </UISModelElement>
  </XMI.extensions>
</XMI>
```

Fin de la définition des éléments

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

4. ETUDE DE L'EXISTANT

4.1 LISTE EXHAUSTIVE DES OUTILS

Voir annexe 1.

4.2 SUPPORT A XMI

A number of UML tools have recently started shipping support for XMI. We hope to create a short list of these tools in this thread.

Note that interoperability testing has been done partially by us for these tools (although we do not plan to do this eventually, for all tools).

- *April, 2001 : Scriptor 2.4 can read an XMI file and use it to generate code, which may be customized.*
- *May, 2001 : MagicDraw UML 4.5 can read and write UNISYS UML XMI.*
- *June, 2001 : Structure Builder 4.5 from Webgain now supports XMI.*
- *June, 2001 : ObjectDomain R3 was announced by ObjectDomain Systems to include XMI import/export.*
- *July, 2001 : Ideogramic UML 1.0 is an innovative new UML tool which uses gesture-based input for true electronic whiteboard modeling sessions. The default file format for this tool is XMI.*
- *August, 2001 : Sparx Systems Enterprise Architect 2.5 has just been released and now supports import and export of models using XMI.*
- *September, 2001 : Boldsoft's Bold for Delphi 3.1 now supports XMI import from other tools.*
- *December, 2001 : Softera's SoftModeler 3.5 has just been released and now supports XMI export.*
- *January, 2002 : Adaptive Arts R3.3 now supports XMI 1.1 with UML 1.3: support for exporting library contents in XMI/UML format, with optional Rose 98 Diagram views*

4.3 SOLUTIONS POTENTIELLES POUR REALISER LES PASSERELLES

Hormis les solutions propriétaires (J, avec objecteering de Softeam, par exemple), nous avons identifié deux types de solutions :

1. L'utilisation d'outils pour fabriquer des "générateurs" (qui prennent en entrée un modèle UML, via XMI) :
   - Scriptor (société française Sodifrance – voir annexe 2)
   - Gen-it (société québécoise Codagen)…
2. L'utilisation d'AGL où on peut rajouter des scripts de génération :
   - Rational Rose (qui est interfacé avec IBM Alphaworks – voir annexe 3)

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

- Together (scripts en Java)…

Dans le chapitre suivant nous allons exposer l'évaluation effectuée de chacune des solutions.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

5. ANNEXE 1 : CATALOGUE DES PRODUITS

Company	Product	Version	Date	Platform	Price
Computer Associates	AllFusion Component Modeler	4 Jan-00	Windows, Unix	n/a	
multiple code generations, object database repository, data modeling integration formerly Paradigm Plus					
Tigris	ArgoUML	0.81	Oct-00	Java VM	$0
open-source project, written in Java, run-time model critique, OCL, XMI					
BoldSoft	Bold for Delphi	3.1	Sep-01	Delphi	$2900
OCL, forward engineering for Delphi, SQL generation, XMI import					
Project Technology	BridgePoint	5 Apr-01	Windows, Unix	n/a	
UML models compiled to executable code, supports Shlaer-Mellor method model verification through animation					
CanyonBlue	Cittera	1 Jun-01	Java VM	n/a	
web-based, collaborative, multi-user tool					
Logic Explorers	Code Logic	Mar-02	Java VM	n/a	
reverse engineer Java class and sequence diagrams, supports JBuilder, JDeveloper					
Honeywell	DOME	5.3	Mar-00	Smalltalk	$0
extensible notations, GNU GPL license, written in Smalltalk! FTP site for exchanging models					
Atos Origin	Delphia Object Modeler (D.OM)	3.2.6	Dec-00	Windows	$0
auto-generation of functional prototypes from UML models, XMI, class and state diagrams, report generation					
Embarcadero	Describe		Jul-01	Windows	n/a
based on GDPro, integrates with leading Java IDE's, EJB Support					
Dia	Dia	0.88	May-01	Linux	$0
gnome Visio-like diagram tool with a UML template, export as SVG!					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Company	Product	Version	Date	Platform	Price
Documentator	Documentator	4	Jan-02	Windows	$89
generate documentation from Rose 2000 to MS Word					
Eldean AB	ESS-Model	2	Oct-01	Windows	$0
generate class diagrams from Java and Kylix code, XMI export, HTML generation open-source, GNU GPL license					
EctoSet	EctoSet Modeller	1.5.4	Apr-02	Windows	$80
Visio-like feel, scripting for code generation					
Elixir Technologies	Elixer CASE	1.2.4	Nov-99	Java VM	$295
auto-generation of sequence diagrams, metrics, OCL, XMI					
Sparx Systems Enterprise	Architect Professional	3.1	Jan-02	Windows	$149
use cases, contracts (pre/post conditions), round-trip engineering for C++, C#, Java, VB.Net. multi-user, project estimation, excellent, free UML tutorial XMI import/export					
Tassc	Estimator / manager	4.1	Mar-01	Windows	$4255
project estimation and scheduling tool based on OO criteria, interfaces to Rational Rose, Select Enterprise, Together Control Center					
Novosoft	FL 0.	5.5	Feb-02	Java VM	n/a
develop Java object persistence from class diagrams Rational Rose add-in, supports OQL, supports major DBMS					
Fujaba	Fujaba	3	Jan-02	Java VM	$0
open-source GNU GPL license, forward and reverse engineering for Java University of Paderborn project					
Hoora	HAT Professional	3.1	Mar-01	Windows	$800
supports HOORA process, C++ forward engineering, report generation requirements management, Rose import					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Company	Product	Version	Date	Platform	Price
Ideogramic	Ideogramic UML, Desktop Edition	2.2	Apr-02	Windows, Linux	$1160
C++ and Java reverse engineering, XMI, image export, documentation generation class, use case, sequence and state diagrams					
Ideogramic	Ideogramic UML, Whiteboard Edition	2.2	Apr-02	Windows	$5195
adds innovative input scheme using gestures on large whiteboards					
Borland	JBuilder Enterprise	6	Mar-02	Java VM	$2999
class and package diagrams, code navigation, Java reverse engineering, refactoring					
Oracle	JDeveloper	9i	Jan-02	Windows	$0
class and activity diagrams, Java round-trip engineering, XMI export					
JSequence	JSequence		Mar-02	Java VM	$199
create sequence diagrams automatically from java source, XMI export					
Object Insight	JVISION	2.1	Oct-01	Java VM	$499
reverse-engineering of Java, integration with Visual Café, HTML generation 9 UML diagrams plus robustness diagram					
Excel Software	Mac A&D Desktop	7.3	Jun-01	MacOS	$1295
Macintosh version of Win A&D					
No Magic	MagicDraw UML Standard	5.1	Mar-02	Java VM	$299
supports all 9 UML 1.3 diagrams, Swing GUI, HTML generation, read Rose models, XMI, SVG, XSLT, EJB					
No Magic	MagicDraw UML Professional	5.1	Mar-02	Java VM	$699
adds forward and reverse engineering for Java, C++, IDL					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Company	Product	Version	Date	Platform	Price
Mega International	Mega Suite	5.2	Oct-00	Windows	n/a
supports Delphi, Forte, Java, VB, XML					
MetaCase Consulting	MetaEdit+	3		Windows, Unix	$4500
multi-user object repository, customizable meta-tool, report generation. Java, C++, Smalltalk, IDL, Delphi, SQL					
Metamill Software	Metamill	2	Nov-01	Windows	$85
low-cost tool, index file based shared repository round-trip engineering for Java, C++, C#					
ModelMaker Tools	ModelMaker	6.1	Dec-01	Delphi	$269
forward and reverse engineering for Delphi, GOF design patterns					
Modelistic	Modelistic	1	Aug-00	Java VM	$299
round-trip engineering for Java					
Novosoft	Novosoft UML Library	1.4	Dec-01	Java VM	$0
open-source library which supports the UML 1.3 metamodel persistence using XMI, integrated with ArgoUML					
OWiS Software, Germany	OTW - Object Technology Workbench	2.4	Jan-00	Windows	n/a
round-trip engineering for Java, C++, support for patterns, repository, data modeling					
Object Plant	Object Plant	3.0.1	Feb-02	MacOS	$35
shareware, class, state, use case diagrams, C++, Java					
Object Domain Systems	Object Domain Standard	3	Nov-01	Java VM	$495
forward and reverse engineering for C++, Java, Python,. Python scripting, educational pricing available					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Company	Product	Version	Date	Platform	Price
OTW Software	Object Technology Workbench Private	2.4	Apr-00	Windows	$795
round-trip engineering for Java, C++, Delphi supports CORBA-IDL, SQL-DDL, patterns, repository, HTML					
Object Domain Systems	Object Domain Professional	3	Nov-01	Java VM	$995
adds multi-user repository, round-trip engineering for Java, HTML generation					
OTW Software	Object Technology Workbench Team	2.4	Apr-00	Windows	$1495
adds team-based repository					
Telelogic	ObjectGeode	4.1	Jun-99	Windows, Unix	n/a
real-time modeling, multiple RTOS targets, generates C, C++					
Softeam	Objecteering Personal Edition	5.1.0	May-01	Windows, Unix	$0
free, base version, includes XMI support					
Softeam	Objecteering Personal Edition / Java	5.1.0	May-01	Windows, Unix	$859
adds round-trip engineering for Java					
Softeam	Objecteering Project Edition	5.1.0	May-01	Windows, Unix	$1569
full-featured product without multi-user repository support					
Softeam	Objecteering Enterprise Edition	5.1.0	May-01	Windows, Unix	$2569
adds parameterized code generation, multi-user repository, data modeling, design patterns, metrics					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Company	Product	Version	Date	Platform	Price
MicroTOOL	ObjectiF	4.6		Windows	n/a
VB scripting, C++, Java, IDL, XMI, integration with JBuilder					
TNI	OpenTool	3.2	Jan-01	Windows, Unix	n/a
forward engineering for C++, Java, Smalltalk, reverse engineering for Java,. multiple documentation generations					
Plastic Software	Plastic	3	Jan-01	Java VM	$297
forward and reverse engineering for Java, HTML generation, model validation					
Gentleware	Poseidon for UML	1.2	Mar-02	Java VM	$0
based on ArgoUML, adds commercial support and services integration with Forte for Java, OCL, SVG					
Sybase	PowerDesigner	9	Dec-01	Windows	$5990
object/relational design using class diagrams, repository support includes use case and sequence diagrams					
ProxySource	ProxyDesigner	1	Dec-00	Windows	$0
publish UML models directly to online forums					
Excel Software	QuickUML	1	Apr-01	Windows	$495
entry-level tool supports a core set of UML diagrams, XMI					
Tri-Pacific Software	Rapid RMA	5.2	Nov-01	Java VM	n/a
simulation and test for real-time UML integrates with Rational's Rose RealTime and I-Logix Rhapsody					
Rational	Rational XDE Professional .NET Edition	2002	Feb-02	Windows	$3595
C#, fully integrated with Visual Studio .NET, patterns support, round-trip engineering					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Company	Product	Version	Date	Platform	Price
Rational	Rational XDE Professional Java Edition	2002	Feb-02	Windows	$3595
C#, fully integrated with IBM Websphere Studio and Eclipse J2EE support, patterns support, round-trip engineering					
Artisan	Real-time Modeler	4.1	Aug-01	Windows	$2495
real-time modeling, multi-user object repository					
Artisan	Real-time Studio Professional	4.1	Aug-01	Windows	n/a
adds round-trip engineering for C, C++, Java, ADA, Spark DOORS integration, state machine animation					
I-Logix	Rhapsody Modeler	4	Sep-01	Windows	$0
real-time, C, C++, single-user, free starter version					
I-Logix	Rhapsody Solo	4	Sep-01	Windows	$895
real-time, C, C++, Java, single-user, XMI					
I-Logix	Rhapsody Development	4 Sep-01	Windows	n/a	
real-time, C, C++, Java, multi-user, XMI					
Rational	Rose Modeler	2002	Jan-02	Windows	$1829
base model					
Rational	Rose Professional	2002	Jan-02	Windows	$2442
adds round-trip engineering, repository support, data modeling - Java, C++, and VB versions sold separately					
Rational	Rose Enterprise	2002	Jan-02	Windows	$4290
adds web publishing, CORBA-IDL - integration w/ ClearCase (version control) and MS VisualStudio (VB, C++ only)					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Company	Product	Version	Date	Platform	Price
Rational	Rose Real Time	2002	Jan-02	Windows	n/a
real-time modeling based on ObjecTime technology					
Sodifrance	Scriptor	2.5	Oct-01	Java VM	$4000
meta-generator providing the capability to build your own specific code generator reads XMI files					
Aonix	Select/Enterprise			Windows	n/a
component repository, data modeling integration,. round-trip engineering for C++, Java, Forte, VB					
Magna Solutions	Silverrun Java Designer	1.1		Java VM	$0
forward and reverse engineer Java code to class diagrams					
Adaptive Arts	Simply Objects Standard	3.3.3	Mar-02	Windows	$269
forward engineering for Delphi, Smalltalk, Eiffel, Java, C++, Corba, VB. XMI, export diagrams as jpeg, png					
Adaptive Arts	Simply Objects Professional	3.3.3	Mar-02	Windows	$1999
adds use case and interaction diagrams, report generator, multi-user					
Softera	SoftModeler Standard	3.5	Dec-01	Java VM	$245
base model					
Softera	SoftModeler Professional	3.5	Dec-01	Java VM	$495
adds EJB support, round-trip synchronization, XMI export					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Company	Product	Version	Date	Platform	Price
Softera	SoftModeler Enterprise	3.5	Dec-01	Java VM	$995
Adds multi-user, shared repository, model simulation sequence-diagram animation					
Aonix	StP/UML	8.2	Jun-01	Windows, Unix	$2500
multi-user repository, DOORS integration, report generation Forte, Smalltalk, Java, C++					
WebGain	StructureBuilder Enterprise	4.5.4	Jun-01	Java VM	$4995
round-trip engineering, HTML generation, component of WebGain Studio EJB support, XMI, round-trip engineering of sequence diagrams (unique)					
Popkin	System Architect	8.5	Oct-01	Windows	n/a
round-trip engineering for Java, C++, VBA. data modeling, Microsoft repository support, scripting, DOORS support					
Telelogic	Tau UML Suite	4.3	Sep-01	Windows	n/a
real-time modeling, UML to SDL translation, XMI. acquired COOL:Jex from Sterling Software, DOORS from QSS					
TogetherSoft	Together CommunityEdition	6 Apr-02	Java VM	$0	
simultaneous round-trip engineering for Java, C++, class diagrams only					
TogetherSoft	Together Solo	6	Apr-02	Java VM	$3495
adds complete UML diagram support, HTML generation, code debugger, refactoring					
TogetherSoft	Together Control Center	6	Apr-02	Java VM	$5995
adds EJB development and deployment support, GOF design patterns, VB, .NET, C#					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Company	Product	Version	Date	Platform	Price
Pragsoft Corporation	UML Studio	6.2	Feb-02	Windows	$500
forward and reverse engineering for C++, Java, IDL, scripting tools, auto-save - integration tool for VC++/C++, add in for Rational Rose, COM+ code generator					
Unimodeler	Unimodeler	1	Mar-02	Linux	$0
supports all 9 UML diagrams, GTK (Gnome) based, postscript printing					
Microsoft Visio	Visio 2002 Professional	2002	Mar-01	Windows	$499
C++, VB reverse engineering, MS Visual Studio					
Visual Object Modelers	Visual UML Standard Edition	2.9	Mar-02	Windows	$495
C++, C# and Java code generation and reverse engineering, data modeling					
Visual Object Modelers	Visual UML Standard Edition for VB	2.9	Mar-02	Windows	$795
adds VB support, VB round-trip engineering					
Visual Object Modelers	Visual UML Plus Edition for VB	2.9	Mar-02	Windows	$995
includes VBA, MS Repository 2.0 support					
Microsoft	Visual Studio .NET Enterprise Architect	1	Feb-02	Windows	$2500
supports 8 UML diagrams through Visio integration					
Visual Paradigm	Visual Paradigm for UML	1	Dec-01	Java VM	n/a
Java forward engineering, HTML and PDF generation site has feature demo using viewlets					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

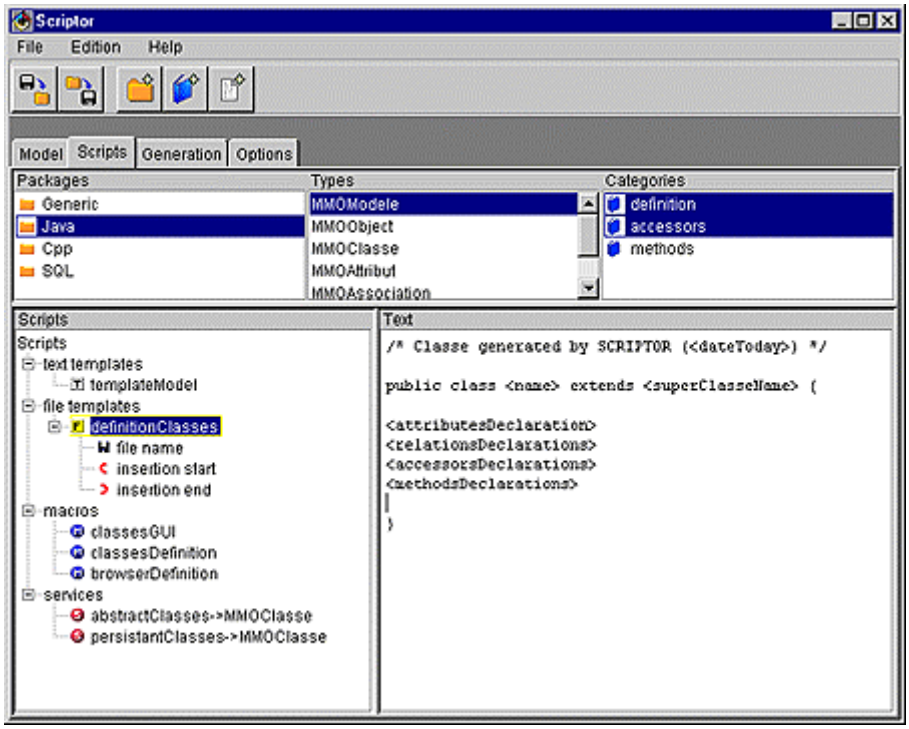
Company	Product	Version	Date	Platform	Price
Excel Software	Win A&D Standard	3.3	Jun-01	Windows	$495
CRC card support, component modeling					
Excel Software	Win A&D Desktop	3.3	Jun-01	Windows	$1295
adds forward-engineering for Java, Delphi, C++, scripting, state models					
Microgold Software	WithClass Professional	2000	Mar-01	Windows	$495
multiple code generation, Python scripting! now supports C#					
Microgold Software	WithClass Enterprise	2000	Mar-01	Windows	$800
adds VBA support for scripting					
Tojosoft	d.vine	1	Apr-02	Delphi	$0
reverse-engineering of Object Pascal (Delphi)					
Kennedy-Carter	iUML	2	Jan-01	Windows, Unix	$3000
produce executable UML models using a formal action language, includes code generator and simulator tools					
Mountfield Computers	mDes	5.2.4	Feb-02	Java VM	$65
supports all 9 UML 1.3 diagrams, free for non-commercial use forward and reverse engineering for Java, Python XMI export, HTML generation, SVG					

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

6. ANNEXE 2 : DESCRIPTION DE L'OUTIL SCRIPTOR

SCRIPTOR est un environnement de développement de générateurs. Il facilite la création et la maintenance de générateurs spécifiques[1].

SCRIPTOR a été développé par le groupe Sodifrance dans le cadre de grands projets de migration vers les nouvelles technologies. Sur ces projets, l'utilisation d'outils de modélisation et d'architectures cibles variées ont nécessité une solution ouverte et extensible.



Au sein de l'outil SCRIPTOR, le méta-modèle UML est exprimé avec le formalisme de sNets. Il en est de même pour les modèles UML importés dans SCRIPTOR à partir des différents outils de modélisation. Ce formalisme permet à l'outil d'être complètement indépendant du méta-modèle sous-jacent et de profiter de tous les composants développés autour de ce formalisme comme l'import et l'export au format XMI par exemple.

De plus, les différents scripts de l'outil sont rattachés aux entités de type "EntityType" du méta-modèle et le mécanisme de gestion des scripts est totalement indépendant du méta-modèle tandis que les scripts peuvent utiliser la notion d'héritage définie pour les entités de ce type et profiter du polymorphisme.

Acceptant des modèles issus de différents AGL et facilement adaptable à tout type d'outil propriétaire, il permet aussi bien :

- D'engendrer du code : génération de composants EJB, de code JSP, d'implémentations Java, de structure de tables, de code Smalltalk, C++…

[1] Annexe rédigée à partir de la fiche disponible sur le site web de SodiFrance http://www.sodifrance.fr/

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

- D'engendrer de la documentation
- De transformer des modèles dans le cadre d'un changement d'AGL
- D'établir des ponts entre logiciels…

6.1 LES ATOUTS D'UN GENERATEUR DE CODE

Un générateur de code intervient essentiellement lors de la phase de conception d'une application. Il offre un certain nombre d'intérêts dont les principaux sont exposés dans les paragraphes suivants.

6.1.1 Réduction du temps de développement

Il arrive fréquemment que l'on ait à écrire du code systématique sur un grand nombre d'entités (les accédants à toutes les classes par exemple). Le code est alors presque identique à chaque fois : seuls quelques caractères changent (le nom de l'entité par exemple). Lorsque ce type de code est à reproduire un très grand nombre de fois, le fait de le générer permet de gagner un temps non négligeable.

6.1.2 Qualité

Lorsque le code est généré, on réduit considérablement le nombre d'erreurs. En effet, on élimine toutes les fautes de frappes, toutes les fautes de copier-coller (technique très utilisée sans générateur), tous les oublis… Tous ces types d'erreurs ne sont pas forcément visibles au premier coup d'œil, d'où le risque d'en laisser dans le code final.

Par contre dans le cas de la génération, si l'on fait une faute lors de l'écriture du script, la faute est généralement répercutée sur toutes les entités : la détection est donc rapide. Une correction du script et une génération complète permettent d'éliminer l'erreur. Enfin, si le code est généré, on assure un format unique de présentation, ce qui permet de rendre le code plus lisible, et donc plus facile à maintenir.

6.1.3 Extensibilité

Lorsque l'on commence la conception, on n'a pas nécessairement toutes les bonnes idées dès le départ. Deux possibilités se présentent alors au concepteur :

- commencer rapidement le codage, au risque de faire de mauvais choix, et d'avoir ensuite à réécrire une partie du code
- repousser le début du codage tant que tous les concepts n'ont pas été complètement validés, au risque de retarder les autres phases du projet

Lorsque l'on dispose d'un générateur de code, on est moins confronté à ce dilemme, puisqu'une idée de conception qui arrive tardivement peut être facilement intégrée, et ce quelque soit le nombre d'entités impactées. On peut ainsi commencer la conception même si tous les concepts ne sont pas complètement arrêtés, quitte à revenir ultérieurement sur certaines parties du code. De cette manière, il est possible de livrer rapidement une première version, avec laquelle il sera plus aisé de détecter les éventuels problèmes

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

6.1.4 Continuité entre l'analyse et la conception

Dans tout projet de taille significative, l'analyse constitue une part importante de la charge. Cette analyse a pour but, entre autres, d'obtenir un modèle de conception qui va guider les développeurs dans leur tâche de réalisation. Tout serait parfait si, une fois l'analyse terminée, on était certain d'avoir pensé à tout et d'avoir tout modélisé comme il le fallait. Or, il est rare qu'une fois la phase de conception commencée, on ne revienne plus sur la phase d'analyse. En effet, outre les oublis éventuels, on n'est jamais complètement à l'abri d'une évolution des spécifications et des attentes de futurs utilisateurs.

Lorsque la conception est commencée, toute évolution implique une modification à la fois du modèle et du code déjà écrit. Cette double intervention a un coût très important car c'est une opération fastidieuse qui prend du temps et de l'énergie. Lorsque les délais sont courts, le risque existe que les modifications ne soient plus effectuées que sur le code et qu'elles ne soient pas répercutées sur le modèle. Dans ce genre de cas (très fréquents) le modèle est petit à petit délaissé, puis complètement abandonné lorsque les différences sont trop importantes.

Avec un outil de génération, c'est le modèle qui reste la référence tout au long du cycle de vie du projet. En effet, les modifications peuvent être reportées sur le modèle (et non plus directement dans le code), à partir duquel on peut toujours engendrer le code nécessaire. De cette façon, le modèle n'est plus seulement une représentation de ce qui doit être conçu, mais devient aussi une représentation de ce qui est conçu (voire de ce qui a été conçu, ce qui est très utile lors de la phase de maintenance)...

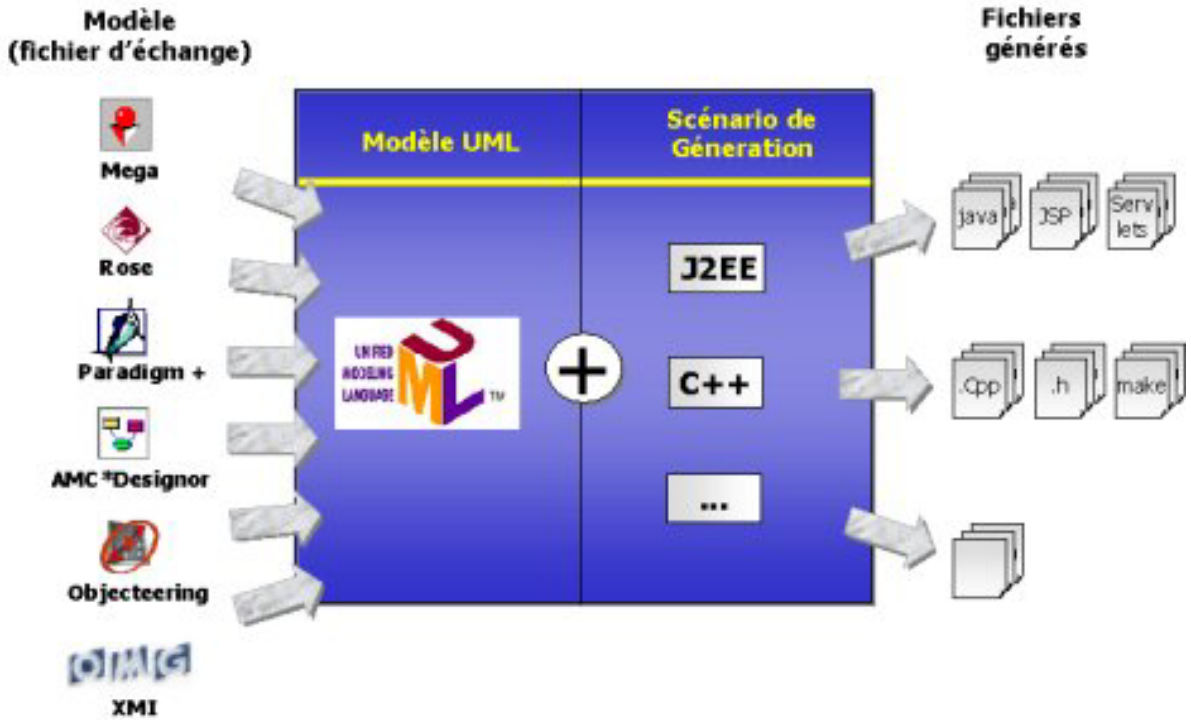
6.1.5 Réactivité

Globalement l'utilisation d'un générateur de code permet une meilleure réactivité lors de la phase de conception.

6.2 UNE ARCHITECTURE OUVERTE

L'architecture de SCRIPTOR est complètement ouverte.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.



Elle est entièrement articulée autour d'un besoin : celui de mettre en relation d'une part un modèle résultant de la phase d'analyse, et d'autre part des scripts de génération qui traduisent des choix de conception

Pour être importé dans SCRIPTOR, le modèle peut être conçu à l'aide des principaux AGL UML du marché :

- ROSE (Rational)
- MEGA Development (Mega International)
- OBJECTEERING (Softeam)
- RHAPSODY (Ilogix)
- PARADIGM PLUS (Platinum)
- POWER AMC (Sybase)
- Et tout AGL compatible XMI.

Le type des fichiers générés est complètement ouvert et laissé au libre choix de l'utilisateur :

- Code source (C++, Java, Delphi, Ada, XML...)
- Documentation (liste des classes du modèles, contrôles de cohérences)
- Migration vers un autre AGLI.

6.3 UNE UTILISATION SIMPLE ET EFFICACE

6.3.1 Editeur WYSIWYG

SCRIPTOR consiste à mettre en relation un modèle d'analyse et des scripts de génération.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

Grand atout de SCRIPTOR, son éditeur de scripts est de type WYSIWYG.

Ainsi, il suffit de saisir le code qui doit être généré (dans lequel on insère des noms de macros) au lieu de le programmer. La création des scripts est ainsi grandement facilitée, puisqu'il est possible de les concevoir grâce à des copier-coller à partir d'éléments de code déjà écrits et validés.

Text

```
public [[javaType]] get[[JavaName]]() {
    return [[javaName]];
}
public void set[[JavaName]]([[javaType]] uneValeur) {
    [[javaName]] = uneValeur;
}
```

Exemple de génération Java

6.3.2 JAVA pour accéder au modèle

Conférer un caractère générique à un script, en introduisant des éléments variables, comme le nom de la classe par exemple, nécessite l'utilisation d'un langage de commande.

SCRIPTOR permet de définir des «macros» écrites en langage JAVA.

L'utilisation de Java permet de profiter des possibilités offertes par ce langage, et évite l'apprentissage d'un langage propriétaire de type BASIC, solution généralement proposée par les AGL.

Cycles itératifs

Les fichiers générés peuvent être enrichis par du code spécifique écrit à la main. SCRIPTOR peut conserver ce code lors des générations ultérieures

6.3.3 Organisations en "packages"

SCRIPTOR propose une organisation pour faciliter la maintenance et la réutilisation.

Chaque script et chaque macro, en plus d'être rattaché à un type d'objet, doit être rattaché à un package.

Dans un package, il est possible de regrouper des scripts et des macros rattachés à des types d'objets différents.

Le but est de pouvoir organiser dans des entités différentes les scripts écrits pour des besoins différents. Cela permet d'organiser les scripts par niveaux : des plus génériques aux plus spécifiques.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

6.3.4 Scenarii de génération

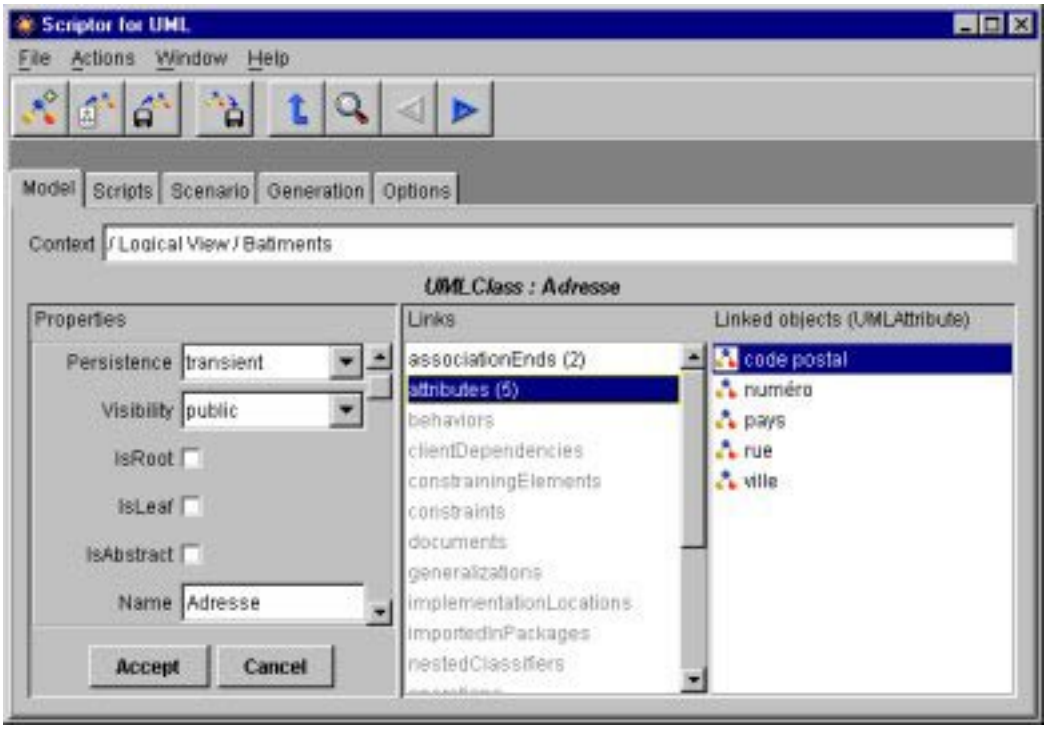
Le plus souvent la génération de tous les fichiers résulte de l'application de plusieurs scripts sur des objets différents du modèle.

Par exemple un premier script sur le modèle, un deuxième script sur toutes les classes persistantes et un troisième script sur toutes les classes non persistantes. Pour éviter d'avoir à effectuer manuellement les trois générations successives, avec tous les risques d'erreur possibles (oubli d'une génération, erreur dans le choix d'un scripts, …), il est possible de créer un scénario qui définit une succession de générations.

Un scénario peut lui-même servir à construire d'autres scénarios.

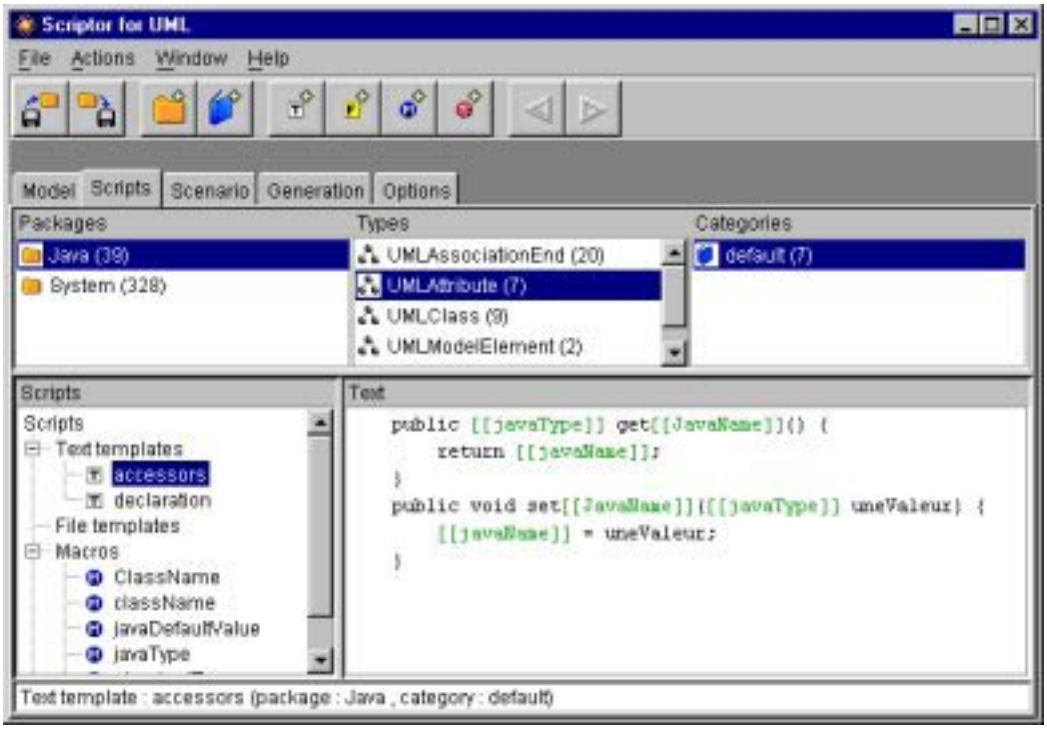
6.4 UNE INTERFACE CONVIVIALE

6.4.1 Navigation dans le modèle

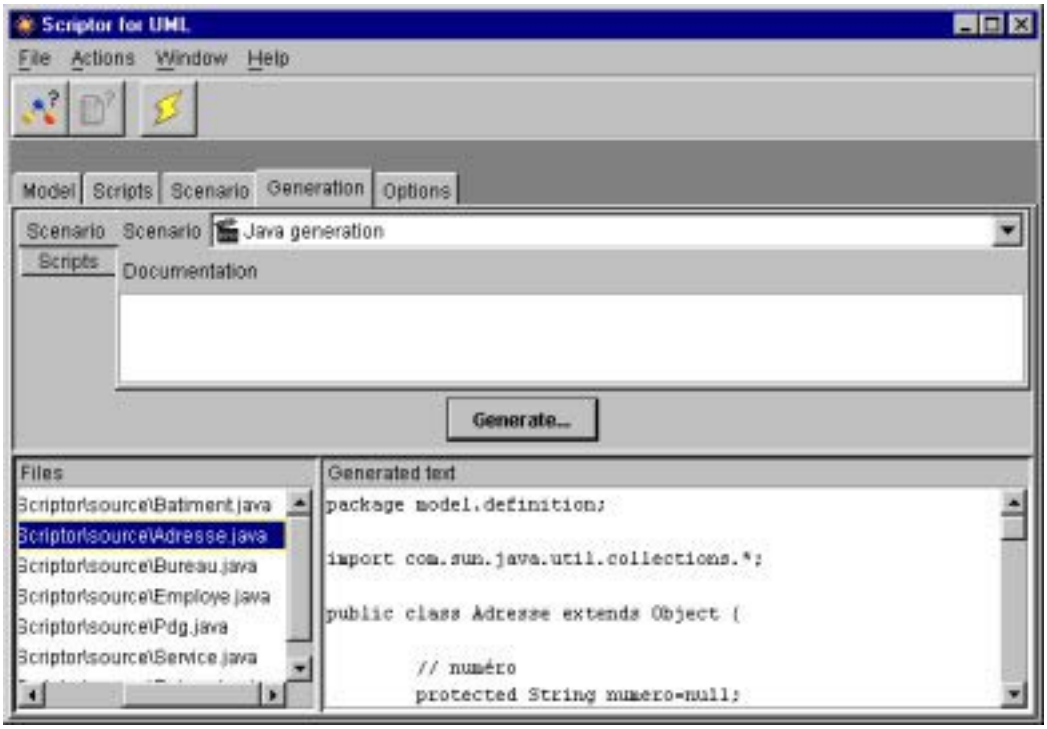


© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

6.4.2 Edition des scripts



6.4.3 Génération



© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

7. ANNEXE 3 : "XMI TOOLKIT" DE "IBM ALPHAWORKS"

7.1 THE XMI STANDARD

The XML Metadata Interchange Format (XMI) specifies an open information interchange model that is intended to give developers working with object technology the capability to exchange programming data over the Internet in a standardized way[2].

XMI is an accepted industry standard that combines the benefits of the Web-based XML standard for defining, validating, and sharing document formats on the Web with the benefits of the object-oriented Unified Modeling Language (UML). UML itself is a specification of the Object Management Group (OMG) that provides application developers with a common language for specifying, visualizing, constructing, and documenting distributed objects and business models.

By using an industry standard for storing and sharing object programming information, development teams using tools from multiple vendors can collaborate on applications. The XMI standard will allow developers to leverage the Web to exchange object-oriented data among tools, applications, and repositories, and to create secure, distributed applications built in a team development environment.

7.2 XMI TOOLKIT - OVERVIEW

The XMI Toolkit for VisualAge for Java utilizes the XML-based standard, XMI, to perform transformations on different forms of programming and modeling information. XMI is an open standard for general purpose information interchange and messaging. XMI can be applied to a wide variety of information exchanges and transformations. This XMI Toolkit implementation specializes in interchange of UML design information and Java source code. The Toolkit consists of a graphical user interface and command line programs that are used to perform these conversions. Additionally, given an XMI document, the XMI Toolkit can generate a corresponding DTD.

For example, the XMI Toolkit for VisualAge for Java allows you to develop applications by jointly using Rational Rose and IBM VisualAge for Java. In a typical scenario, you first develop an object-oriented analysis model for your application using the Rational Rose visual modeling tool. Next, you use the XMI Toolkit to generate Java code corresponding to the model. The generated Java code can then be imported into IBM VisualAge for Java, where the implementation of the application's class methods can be completed.

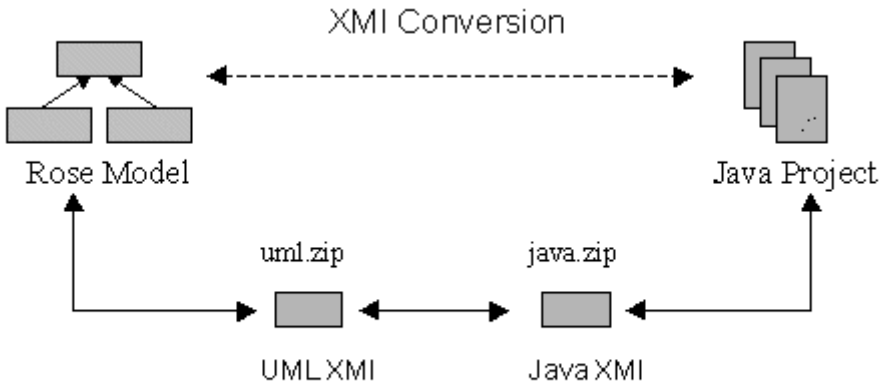
During the development phase in VisualAge for Java, it may be necessary to make structural modifications to the original analysis model you created using Rose. For example, you might need to add or delete some classes, methods, or fields, or to change the structure of the class inheritance hierarchy. The XMI Toolkit provides the capability to create an updated version of the analysis model that reflects the changes made during development in VisualAge for Java. In this way, the analysis model can be kept in synch with the application source code.

---

[2] Annexe rédigée à partir d'informations recueillies sur le site web de IBM AlphaWorks http://www.alphaworks.ibm.com et de la documentation associée à la bibliothèque XMI Toolkit.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

The diagram below shows the steps involved in the conversion process between a Rose model and the generated Java source. The broken line in the upper portion of the diagram shows how you can conceptually envision the conversion process between a Rose analysis model and the corresponding Java source files in a Java project. The solid lines in the lower portion of the diagram provide a more detailed flow of how the XMI conversion is actually carried out.



The steps are as follows :

3. The Rose model is converted into one or more XMI files that contain the UML XMI representation of the model's structure. The UML XMI files are packaged into a single file named “uml.zip”. This file is written to the same directory where the corresponding Rose “.mdl” file is stored.
4. The XMI Toolkit then maps the UML XMI representation into an equivalent representation in Java XMI. The corresponding Java XMI files are stored in a single file named “java.zip”. This file is written to the top level directory ("project directory") under which the Java project source files are to be written.
5. Finally, the XMI Toolkit generates the Java source files corresponding to the representation in Java XMI.

The process also works in reverse. If you have already implemented an application in Java, you can use the XMI Toolkit to derive a corresponding Rose model. You can then work with the generated model file to document or enhance the application's design.

Important!

When roundtripping between VisualAge for Java and Rational Rose, you must specify the same locations (directory and “.mdl” file name for Rose, and Java project directory for Java) for the input and output fields when going back and forth. The file names must stay the same; consequently, the files will be overwritten each time conversion occurs. This is required since the XMI information in the “uml.zip” and “java.zip” files written during the previous conversion is needed to perform the correct transformation on the conversion back.

Therefore, we recommend that you always keep a backup copy of your model (i.e. the “.mdl” and .cat files).

7.3 XMI TOOLKIT BROWSER AND SMARTGUIDE

The XMI Toolkit user interface consists of two components :

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

- The XMI Toolkit Browser and
- The XMI Toolkit SmartGuide

You use the XMI Toolkit Browser to examine the XMI conversion mapping between a Rose model and a Java project. The Browser displays the UML XMI representation of your Rose model in the “uml.zip” file corresponding to that model; it displays the Java XMI representation of a Java project in the “java.zip” file corresponding to that project. The Browser can also be used to show differences between successive versions of the UML XMI representation of a Rose model, or the Java XMI representation of a Java project.

The XMI Toolkit SmartGuide is launched from the Browser whenever you initiate an XMI conversion between a Rose model or Java project. The SmartGuide provides a sequence of easy-to-use panels that assist you in organizing and specifying the details of the conversion you want to perform.

7.3.1 The XMI Toolkit Browser

Using the XMI Toolkit Browser view, you can :

6. Launch the XMI Toolkit SmartGuide to perform an XMI conversion between a Rose model and a Java project.
7. Examine the XMI mapping between a Rose model and a Java project.
8. Compare the differences between two versions of the UML XMI representation of a Rose model.
9. Compare the differences between two versions of the Java XMI representation of a Java project.
10. Generate a DTD.

The Toolkit Browser contains a side-by-side display of two tree views. Each tree view displays the hierarchical structure of either the UML XMI representation of a Rose model or the Java XMI representation of a Java project. You use both tree views together either to examine the mapping between a Rose model and its corresponding Java project, or to compare the differences between the two most recent versions of either UML XMI or Java XMI.

When you use the Browser to view an XMI conversion mapping between Rose and Java, one tree displays the Rose model in UML XMI, while the other tree displays the Java project in Java XMI. The tree on the left always contains the XMI representation of the input to the conversion, and the tree on the right contains the XMI representation of the output.

When you are viewing differences between the UML XMI representation of a Rose model with the previous version, both trees display UML XMI information. Additionally, a panel is displayed beneath the two tree views that contains a table of the differences between the two versions of the UML XMI. The leftmost tree always displays the newer version of the UML XMI. It reflects the XMI information contained in the “uml.zip” file located in the Rose model directory. The rightmost tree displays XMI information contained in the previous version of the “uml.zip” file, which is named “uml_old.zip”.

Similarly, when you are viewing differences between the Java XMI representation of a Java project with the previous version, both trees display Java XMI information. The panel beneath the two tree views contains a table of the differences between the two versions of the Java XMI. The leftmost tree always displays the newer version of the Java XMI. It reflects XMI information

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

contained in the “java.zip” file stored in the Java project directory. The rightmost tree displays XMI information contained in the previous version of the “java.zip” file, which is named “java_old.zip”.

7.3.2 The Browser menubar

The Browser menubar has the following four menu choices:

- File
- View
- Window
- Help.

The File menu is used to perform an action within the Toolkit. From the File menu, you can select menu items to :

- Perform an XMI conversion between Rose and Java
- Show the UML/Java mapping created during an XMI conversion between Rose and Java
- Show the differences between two versions of UML XMI
- Show the differences between two versions of Java XMI
- Generate a DTD
- Set the Browser preferences
- Exit the Browser

The View menu contains selections with which you can specify the size and location of the Browser window panes. Additionally, the view menu contains a filter that you can use to specify what kinds of differences you want to view when comparing two different versions of XMI. Specifically, from the View menu, you can select menu items to :

- Maximize or minimize the selected Browser pane.
- Detach the selected pane from the Browser.
- Return a detached pane to the Browser (from the View menu of the detached pane).
- Set the differences filter to include or exclude categories of differences when comparing two versions of XMI.
- Expand or collapse trees in the Browser tree views.

When you detach a pane from the Browser, the View menu for the detached pane will have a Return to the parent page menu item. You use this to return the detached pane to its former position in the Toolkit Browser.

The Windows menu contains a list of the windows associated with the XMI Toolkit. The list includes the Browser itself, and any panes that have been detached from the Browser.

The Help menu contains an item to view general information about the XMI Toolkit.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.

7.3.3 Setting preferences

You can use the Application Preferences dialog to control certain aspects of the XMI Toolkit appearance and behavior. Use the **File > Preferences** menu item to bring up the Application Preferences dialog.

The Application Preferences dialog has the following areas :

- **General** Allows you to set preferences on the appearance of the panes in the Browser. For example, you can specify if you want title bars shown with the panes, or if you want the active pane to be highlighted.
- **Appearance** Allows you to control the general look-and-feel of the Browser by selecting from several Family and Platform options.
- **Toolbar** Allows you to specify the positioning and behavior of the Browser toolbar. Additionally, you can specify aspects of the appearance of the toolbar icons.

© Ce document est la propriété exclusive de **CS Cisi**. Il ne peut être ni reproduit ni communiqué à un tiers sans autorisation préalable.