

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

Suivi des versions-révisions et des validations du document.			
<p>Ce document annule et remplace tout document diffusé de version-révision antérieure.</p> <p>Dès réception de ce document, les destinataires ont pour obligation de détruire les versions-révisions antérieures, toutes les copies, et de les remplacer par cette version.</p> <p>Si les versions-révisions antérieures sont conservées pour mémoire, les destinataires doivent s'assurer qu'elles ne peuvent être confondues avec cette présente version-révision dans leur usage courant.</p>			
Version.	Date.	Auteurs.	Création, modification ou validation.
A	23 nov. 2003.	JPD.	Création.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

1 Tables

1.1 Table des matières

1	Tables	2
1.1	Table des matières	2
1.2	Table des illustrations	3
2	Références	4
2.1	Glossaire	4
2.2	Ressources	4
3	Introduction	5
3.1	Objet du document	5
3.2	Audience	5
3.3	Pré-requis	5
4	Interaction avec l'environnement	6
4.1	Description	6
4.2	Paramètres	6
4.3	Particularités	6
4.3.1	Compilation	6
4.3.2	Exécution	6
4.4	Application Program Interfaces	6
5	Choix techniques	7
5.1	Principes	7
5.2	Analyse lexicale	8
5.2.1	Tokens	8
5.2.2	Paquets lexicaux	8
5.2.3	Gestion des analyseurs lexicaux	9
5.2.4	Déclenchement de l'analyse	9
5.3	Analyse syntaxique	10
5.3.1	Règles de grammaire	10
5.3.2	Gestion des analyseurs syntaxiques	12
5.3.3	Déclenchement de l'analyse	12
6	Modèle de données	13
7	Composants techniques	14

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

1.2 Table des illustrations

Texte 1 – Exemple de syntaxe d'un flux d'un l'échange de données	11
Texte 1 – Lecture de l'exemple de syntaxe d'un flux d'un l'échange de données.....	11
Diagramme 2 – Modèle physique des données publiques du module <i>Up ! Analyzer</i>	13
Tableau 3 – Glossaire du modèle physique des données publiques du module <i>Up ! Analyzer</i>	13
Tableau 4 – Composants techniques du module	14

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

2 Références

2.1 Glossaire

Liste des définitions des termes employés.	
Ce tableau recense tous les termes, les concepts particuliers ainsi que les abréviations employés dans ce document.	
Terme, concept, abrégé.	Définition du terme, du concept ou de l'abréviation.
Paquet lexical	Voir page 8.
Règle d'analyse	Voir page 10.
Token	Voir page 8.
Grammaire	Voir page 10.

2.2 Ressources

Liste des documents applicables et en référence.		
Un document est applicable à partir du moment où son contenu est validé et que l'activité ou le projet fait partie de son périmètre d'application. Il est obligatoire d'appliquer son contenu.		
Un document est en référence à partir du moment où son contenu n'est pas validé ou que l'activité ou le projet ne fait partie de son périmètre d'application. Il est recommandé d'appliquer son contenu mais cela n'est pas obligatoire.		
Un document applicable est indiqué par A1, A2, A3 , etc. Un document en référence est indiqué par R1, R2, R3 , etc.		
Index.	Nom du document.	Commentaire.
A1	UpComp-Plan Qualité-000005	Méthode documentaire.
A2	UpComp-Plan Qualité-000006	Processus de management de projet.
A3	UpComp-Plan Qualité-000046	Méthode de spécification technique d'un module.
A4	UpComp-UspAna-000002	Plan documentaire du projet.
A5	UpComp-UpsVm-000003	Plan de programmation.
A6	UpComp-UpsVm-000004	Programmation en C-- .
R7	http://www.up-comp.com	Site Internet d' Up ! Application System .
A8	UpComp-UpsKrn-000003	Plan d'écriture d'un module.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

3 Introduction

3.1 Objet du document

L'objet de ce document est de décrire le contenu technique du module logiciel **Up ! Analyzer** pour le projet **Up ! Application System**.

Ce document est rédigé et approuvé par la **Maîtrise d'Oeuvre (MOE)**.

3.2 Audience

Ce document s'adresse aux :

- **Directeurs de projets et chefs de projets.**
Pour la compréhension du module technique.
- **Ingénieurs de développement.**
Pour savoir comment est conçu le module technique.

Pour aider ces personnes à remplir le document **Spécification technique d'un module**, leur manager et la cellule de support projet se tiennent à leur disposition.

3.3 Pré-requis

Le pré-requis est la connaissance des documents suivants :

- **Méthode documentaire** [A1].
- **Processus de management de projet** [A2].
- **Méthode de spécification technique d'un module** [A3].

Nous rappelons que tous les documents applicables ou référencés pour le projet **Up ! Application System** sont tracés dans le **Plan documentaire** [A4].

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

4 Interaction avec l'environnement

4.1 Description

L'objet du module logiciel *Up ! Analyzer* est de d'analyser des flux en vue de reconnaître :

- **Des éléments lexicaux – les mots d'une phrase.**
Conformément à un référentiel préétabli.
- **Des éléments syntaxiques – les enchaînements des mots en une phrase.**
Conformément à une grammaire préétablie.

Plusieurs analyseurs lexicaux et syntaxiques peuvent coexister. Ils peuvent être indépendants ou interdépendants.

Les référentiels et les grammaires sont construits dynamiquement. Ils peuvent être également adaptés dynamiquement au cours de l'analyse.

4.2 Paramètres

Up ! Analyzer ne possède aucun paramètre.

4.3 Particularités

4.3.1 Compilation

Néant.

4.3.2 Exécution

Néant.

4.4 Application Program Interfaces

Néant.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

5 Choix techniques

5.1 Principes

Un analyseur est un automate traitant des flux devant correspondre à des :

- **Modèles appelés éléments lexicaux.**
Il s'agit par exemple de mots, de nombres, de signes de ponctuation, etc. dans une phrase.
- **Enchaînements de modèles appelés éléments syntaxiques.**
Il s'agit par exemple de phrases indépendantes.
- **Enchaînements de phrases appelés éléments sémantiques.**
Il s'agit par exemple de phrases conjointes.

Up ! Analyzer est un analyseur générique dynamique pour les flux **Unicode 2.0** pour les phases lexicales et syntaxiques. Quand un élément est reconnu, une fonction de rappel spécifique est appelée. Cette dernière communique à **Up ! Analyzer** le résultat de son traitement selon la convention définie par l'énuméré **EnuUpsAnaAction** :

- **AC_Succes.**
L'analyse continue jusqu'à l'atteinte de la fin du flux.
- **AC_ArreterRegle.**
Une erreur a été détectée, ce qui nécessite d'arrêter de traiter la règle de syntaxe en cours. **Up ! Analyzer** tentera de synchroniser l'analyse sur une autre règle en sautant une partie du flux.
- **AC_ArreterTout.**
Une erreur a été détectée, ce qui nécessite d'arrêter l'analyse en cours.
- **AC_ReprendreSansErreur.**
L'analyse devrait continuer jusqu'à l'atteinte de la fin du flux. Cependant, la fonction de rappel demande d'arrêter de traiter la règle de syntaxe en cours. **Up ! Analyzer** tentera de synchroniser l'analyse sur une autre règle en sautant une partie du flux.
- **AC_ArreterToutSansErreur.**
L'analyse devrait continuer jusqu'à l'atteinte de la fin du flux. Cependant, la fonction de rappel demande d'arrêter l'analyse en cours.
- **AC_SuccesFinal.**
L'analyse est terminée et la fin du flux doit être atteinte.

En cas d'erreur bénigne correspondant soit à **AC_ArreterRegle** ou soit à **AC_ReprendreSansErreur**, **Up ! Analyzer** tente de corriger l'erreur à la volée. Il communique les modifications apportées à une fonction de rappel spécifique, pour laquelle la convention de communication est définie par l'énuméré **EnuUpsAnaCorrection** :

- **AC_Ajouter.**
Ajoute le modèle lexical manquant.
- **AC_AjouterEntier.**
Ajoute le nombre entier manquant.
- **AC_AjouterReel.**
Ajoute le nombre réel manquant.
- **AC_AjouterChaineGuillemets.**
Ajoute la chaîne de caractères entre caractère **guillemet "** manquante.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

- **AC_AjouterChaineApostrophes.**
Ajoute la chaîne de caractères entre caractère **apostrophe '** manquante.
- **AC_Modifier.**
Modifie le modèle lexical incorrect.
- **AC_Supprimer.**
Supprime le modèle lexical inutile.

5.2 Analyse lexicale

Cette section décrit comment les analyseurs lexicaux sont gérés par **Up ! Analyzer**.

Les différentes alternatives possibles sont les suivantes :

- Utiliser un analyseur lexical externe.
Par exemple, **Lex**.
- Utiliser un analyseur lexical propriétaire.
La seconde alternative a été retenue pour la raison suivante :
- **Besoin de plusieurs analyseurs en simultané.**
Le code en langage **C** produit par **Lex** ne le permet pas. Il faut le transformer manuellement.
- **Besoin d'analyseurs dynamiques.**
Pour **Up ! 5GL** qui est extensible et auto-défini. Le code en langage **C** produit par **Lex** ne le permet pas.
- **Fonctionnement en multi-thread.**
Le code en langage **C** produit par **Lex** ne le permet pas.
- **Portabilité complète.**
Le code en langage **C** produit par **Lex** l'est difficilement puisqu'il fait des calculs d'adressage.
Compte-tenu de ces raisons, il n'est pas possible de changer de choix.

5.2.1 Tokens

& Un analyseur lexical reconnaît des modèles auxquels des identifiants numériques sont assignés, appelés **tokens**. Il y a un identifiant différent par modèle.

Un **token** a une position dans une ligne donné par l'énuméré **EnuUpsAnaPositionToken** :

- **PT_DebutLigne.**
Le modèle doit être placé en début de ligne.
- **PT_FinLigne.**
Le modèle doit être placé en fin de ligne.
- **PT_Libre.**
Le modèle peut être placé n'importe où dans la ligne.

5.2.2 Paquets lexicaux

Au fur et à mesure de la progression de l'analyse, de nouveaux **tokens** sont créés correspondant aux nouvelles définitions posées. Cependant, ces définitions peuvent n'être valides que dans un contexte particulier – par exemple une variable locale à une procédure à une fonction.

& Afin de faciliter la gestion des contextes, les définitions sont regroupées dans des **paquets lexicaux** qui peuvent être sélectionnés ou non. Quand ils sont sélectionnés, les définitions qu'ils regroupent sont reconnues.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

5.2.3 Gestion des analyseurs lexicaux

Un nouvel analyseur lexical est créé par l'appel à l'**Application Program Interface (API) AjouterAnalyseurLexical**. Un analyseur lexical existant est détruit par l'appel à l'**Application Program Interface (API) SupprimerAnalyseurLexical**.

Voici les **Application Program Interface (API)** pour gérer les **tokens** :

- **AjouterSeparateurLexical.**
Ajoute un nouveau séparateur lexical. Il peut être supprimé ultérieurement par l'appel à **SupprimerSeparateurLexical**.
- **AjouterToken.**
Ajoute un nouveau modèle lexical. Il peut être supprimé ultérieurement par l'appel à **SupprimerToken**.
- **UtiliserToken.**
Spécifie qu'un **token** va être utilisé ou non. Il ne peut être supprimé quand il est utilisé.

5.2.4 Déclenchement de l'analyse

Le début de l'analyse via un analyseur particulier s'effectue par l'appel à l'**Application Program Interface (API) DebiterAnalyseLexicale**. Elle se termine par l'appel à l'**Application Program Interface (API) TerminerAnalyseLexicale**.

En cours d'analyse, il est possible de :

- Lire le **token** suivant.
Via l'appel à **LireToken**.
- Lire le numéro de ligne et de caractère courant.
Via l'appel à **LireNumeroLigneColonne**.
- Lire les commentaires.
Via l'appel à **ChangerCommentaireToken**. Il s'agit des commentaires collectés depuis le dernier appel à cette **Application Program Interface (API)**.
- Lire la valeur du contexte lexical.
Via l'appel à **ValeurLexicale**. La valeur est encodée selon la convention définie par l'énuméré **EnuUpsAnaValeurLexicale** :
 - **VL_SymboleCourant.**
Il s'agit d'un symbole dont le libellé est le champ **SymboleCourant** du type **ValeurLexicale**.
 - **VL_Chaine.**
Il s'agit d'un symbole dont le libellé est le champ **Chaine** du type **ValeurLexicale**.
La chaîne de caractères est allouée dynamiquement par **Up! Analyzer**. Si elle est récupérée par l'appelant, alors le champ **Chaine** doit être affecté à **NULL**.
 - **VL_Entier.**
Il s'agit d'un symbole dont le libellé est le champ **Entier** du type **ValeurLexicale**.
 - **VL_Reel.**
Il s'agit d'un symbole dont le libellé est le champ **Reel** du type **ValeurLexicale**.
- Lire directement dans le flux en contournant l'analyseur lexical.
Via l'appel à **LireCaractere**.
- Rejeter un caractère directement dans le flux en contournant l'analyseur lexical.
Via l'appel à **RejeterCaractere**.

M

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

- Sélectionner ou non un paquet lexical.
Via l'appel à **SelectionnerPaquetLexical**.
- Chercher un **token** correspondant à un modèle.
Via l'appel à **ChercherTokenParLibelle**.
- Chercher un **token** correspondant à un modèle.
Via l'appel à **SupprimerTokenParLibelle**.

5.3 Analyse syntaxique

Cette section décrit comment les analyseurs syntaxiques sont gérés par **Up ! Analyzer**.

Les différentes alternatives possibles sont les suivantes :

- Utiliser un analyseur syntaxique externe.
Par exemple, **Yacc**.
- Utiliser un analyseur syntaxique propriétaire.
La seconde alternative a été retenue pour la raison suivante :
- **Besoin de plusieurs analyseurs en simultané.**
Le code en langage **C** produit par **Yacc** ne le permet pas. Il faut le transformer manuellement.
- **Besoin d'analyseurs dynamiques.**
Pour **Up ! 5GL** qui est extensible et auto-défini. Le code en langage **C** produit par **Yacc** ne le permet pas.
- **Fonctionnement en multi-thread.**
Le code en langage **C** produit par **Yacc** ne le permet pas.
- **Portabilité complète.**
Le code en langage **C** produit par **Yacc** l'est difficilement puisqu'il fait des calculs d'adressage.
Compte-tenu de ces raisons, il n'est pas possible de changer de choix.

5.3.1 Règles de grammaire

Un analyseur syntaxique reconnaît un enchaînement de **tokens** qui peut être récurrent. Une partie linéaire de cet enchaînement est déclarée sous forme d'une **règle d'analyse**. L'ensemble des règles d'analyse forme la **grammaire**.

- Une règle d'analyse est composée d'une succession de :
 - **Terminaux.**
Il s'agit soit de :
 - **Un caractère, une constante entière, une constante réelle, une constante chaîne de caractères.**
Ils font partie du flux en tant que tel. Ils sont écrits en **vert et gras** dans l'exemple ci-après.
Quand il ne s'agit pas d'un simple caractère, le terminal est identifié par un **token** défini par **Up ! Analyzer**.
 - **Un champ contenant une information échangée.**
L'information fait partie du flux et elle est reconnue parce qu'elle correspond à un modèle identifié par un **token** défini par l'usager d'**Up ! Analyzer**. Elle est écrite en **marron, gras et italique** dans l'exemple ci-après.
 - **Non-terminaux.**
Ils se dérivent en une ou plusieurs règles. Ils sont écrits en *italique* dans l'exemple ci-après.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

- Deux règles dérivant un même non terminal sont séparées par le caractère **tuyau /** et la dernière règle se termine par le caractère **point-virgule ;** dans l'exemple ci-après. Cette convention est non applicable pour **Up ! Analyzer**.

Voici un exemple de grammaire pour analyser le flux de l'échange de données **Changement des contacts** :

```

DebutFlux :
    ListeDEnregistrements
    ;
ListeDEnregistrements :
    Enregistrement
    | ListeDEnregistrements Enregistrement
    ;
Enregistrement :
    " Nom " , " Prénom " , Téléphone , " e-mail " , " Raison sociale " ,
    Siren , " Batiment " , Numéro , " Rue " , Code postal , " Ville " \n
    ;

```

Texte 1 – Exemple de syntaxe d'un flux d'un l'échange de données

Voici comment se lit cet exemple :

- Le flux est composé d'une liste d'enregistrements.
- Une liste d'enregistrements est soit :
 - Un enregistrement.
 - Une liste d'enregistrements suivie d'un enregistrement.
- Un enregistrement commence par le champ **Nom** écrit entre caractères **guillemet "**, suivi par le champ **Prénom** écrit entre caractères **guillemet "**, etc., suivi par le champ **Ville** écrit entre caractères **guillemet "** et suivi par un saut de ligne.

Texte 2 – Lecture de l'exemple de syntaxe d'un flux d'un l'échange de données

Une règle peut comporter :

- **Des paramètres** – Comme un appel de procédure ou de fonction.
La taille de la zone des paramètres est définie par **TailleValeurParametre**. Elle peut être agrandie par l'appel à l'**Application Program Interface (API)** avant de débiter l'analyse, par l'appel à l'**Application Program Interface (API) ModifierTailleParametreEtape**.
- **Un résultat** – Comme une fonction.
La taille de la zone du résultat est définie par **TailleValeurEtape**. Elle ne peut pas être agrandie.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

5.3.2 Gestion des analyseurs syntaxiques

Un nouvel analyseur syntaxique est créé par l'appel à l'**Application Program Interface (API) AjouterAnalyseurSyntaxique**. Un analyseur lexical existant est détruit par l'appel à l'**Application Program Interface (API) SupprimerAnalyseurSyntaxique**.

Une fois l'analyseur créé, il est nécessaire de déclarer tous les non-terminaux par des appels successifs à **Application Program Interface (API) AjouterNonTerminal**.

Voici les **Application Program Interface (API)** pour gérer les règles de grammaire :

- **AjouterRegle.**
Ajoute une nouvelle règle syntaxique. Elle peut être masquée ultérieurement par l'appel à **ChangerEtatRegle**.
- **AjouterEtapeRegle.**
Ajoute une nouvelle étape à la règle à la suite de l'étape précédente. A défaut, il s'agit de la première étape.
- **ChangerPrioriteRegle.**
Spécifie le niveau de priorité de la règle pour résoudre les ambiguïtés. Plus la valeur de la priorité est élevée, plus la règle est prioritaire.

5.3.3 Déclenchement de l'analyse

Le début de l'analyse via un analyseur particulier s'effectue par l'appel à l'**Application Program Interface (API) AnalyserSyntaxiquement**. Elle se termine automatiquement, soit parce que le flux a été correctement analysé ou soit parce qu'il y a eu une erreur de syntaxe fatale correspondant à **AC_ArreterTout** ou à **AC_ArreterToutSansErreur**.

En cours d'analyse, dans une fonction de rappel associée aux étapes de la règle, il est possible de :

- Lire la valeur lexicale correspondant à une étape.
Via l'appel à **LireZoneLexicale** ou via l'appel à **LireZoneValeur**.
- Lire la valeur des paramètres d'une étape.
Via l'appel à **LireZoneParametre**.
- Lire la zone de la valeur de résultat d'une étape.
Via l'appel à **LireZoneResultat**.
- Lire le numéro de ligne et de caractère courant.
Via l'appel à **LireLigneColonneSource**.
- Lire les commentaires.
Via l'appel à **LireCommentaire**.
- Envoyer une erreur de sémantique.
Via l'appel à **EnvoyerErreur**.

Suite à l'analyse, il est possible de connaître le nombre d'erreurs de syntaxique détectées par l'appel à l'**Application Program Interface (API) NbErreursDeSyntaxe**.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

6 Modèle de données

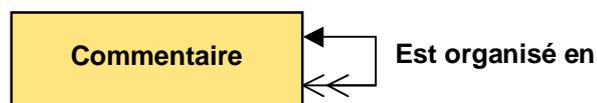
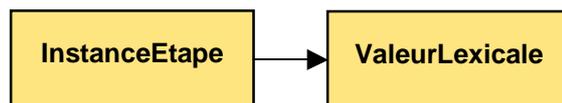


Diagramme 3 – Modèle physique des données publiques du module *Up ! Analyzer*

M Le modèle de données utilise des allocateurs et des tables, à des fins d'optimisation, qui ne sont pas représentés sur le diagramme.

Toutes les entités suivantes sont décrites dans le fichier **uspana.e** :

Entité.	Description.
Commentaire.	Commentaire sur un élément lexical.
InstanceEtape.	Instance d'une étape d'analyse syntaxique.
ValeurLexicale.	Valeur de l'élément lexical reconnu à une étape d'analyse sémantique.

Tableau 4 – Glossaire du modèle physique des données publiques du module *Up ! Analyzer*

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

7 Composants techniques

Le module *Up ! Analyzer* pour le projet *Up ! Application System* est constitué des composants suivants :

Fichiers du module.	
<ul style="list-style-type: none"> Fichier uspana.e – Définition de <i>Up ! Analyzer</i>. Fichier uspana.h – En-tête privé de <i>Up ! Analyzer</i>. Fichier uspana.def – Exportation des symboles pour <i>Windows</i>. 	
Composants.	uspana0.
Description.	
Interface entre <i>Up ! Analyzer</i> et <i>Up ! Module</i> .	
Fichiers.	
<ul style="list-style-type: none"> Fichier uspana0.cpp – Fichier source. Fichier uspana0.h – En-tête privé de uspana0.cpp. Fichier uspana0.e – En-tête protégé de uspana0.cpp. 	
Composants.	uspana1.
Description.	
Moteur d'analyse lexicale générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier uspana1.cpp – Fichier source. Fichier uspana1.h – En-tête privé de uspana1.cpp. Fichier uspana1.e – En-tête protégé de uspana1.cpp. 	
Composants.	uspana2.
Description.	
Moteur d'analyse syntaxique générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier uspana2.cpp – Fichier source. Fichier uspana2.h – En-tête privé de uspana2.cpp. Fichier uspana2.e – En-tête protégé de uspana2.cpp. 	
Composants.	uspana99.
Description.	
Interface entre <i>Up ! Analyzer</i> et <i>Up ! Kernel</i> .	
Fichiers.	
<ul style="list-style-type: none"> Fichier uspana99.cpp – Fichier source. Fichier uspana99.h – En-tête privé de uspana99.cpp. Fichier uspana99.e – En-tête protégé de uspana99.cpp. 	

Tableau 5 – Composants techniques du module

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-UpsAna-000003-A Spécification technique du module UpsAna.doc		

Fin de document

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

Suivi des versions-révisions et des validations du document.			
<p>Ce document annule et remplace tout document diffusé de version-révision antérieure.</p> <p>Dès réception de ce document, les destinataires ont pour obligation de détruire les versions-révisions antérieures, toutes les copies, et de les remplacer par cette version.</p> <p>Si les versions-révisions antérieures sont conservées pour mémoire, les destinataires doivent s'assurer qu'elles ne peuvent être confondues avec cette présente version-révision dans leur usage courant.</p>			
Version.	Date.	Auteurs.	Création, modification ou validation.
A	23 nov. 2003.	JPD.	Création.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

1 Tables

1.1 Table des matières

1	Tables.....	2
1.1	Table des matières.....	2
1.2	Table des illustrations.....	3
2	Références.....	4
2.1	Glossaire.....	4
2.2	Ressources.....	4
3	Introduction.....	5
3.1	Objet du document.....	5
3.2	Audience.....	5
3.3	Pré-requis.....	5
4	Interaction avec l'environnement.....	6
4.1	Description.....	6
4.2	Paramètres.....	6
4.3	Particularités.....	6
4.3.1	Compilation.....	6
4.3.2	Exécution.....	6
4.4	Application Program Interfaces.....	6
5	Choix techniques.....	7
5.1	Principes.....	7
5.2	Extension du noyau linguistique.....	7
5.3	Analyseurs sémantiques.....	8
5.4	Codifications.....	8
5.4.1	Constantes et énumérés.....	8
5.4.2	Types.....	8
5.4.3	Comparaison de types.....	9
5.4.4	Paramètres d'un type.....	9
5.4.5	Contraintes.....	10
5.4.6	Méthodes.....	10
5.4.7	Résultat d'un prototype.....	10
5.4.8	Récurtivité d'un type.....	11
5.4.9	Composition des opérateurs.....	11
5.4.10	Factorisation des opérateurs booléens.....	12
5.4.11	Expressions.....	12
5.4.12	Opérateurs binaires.....	13
5.4.13	Comparaison d'expressions.....	13
5.4.14	Héritages.....	14
5.4.15	Passages de paramètres.....	14
5.4.16	Valeurs par défaut des paramètres.....	14
5.4.17	Variables.....	15
5.4.18	Instructions.....	15
5.4.19	Objets.....	15
5.4.20	Types de fichiers sources.....	16
5.4.21	Implémentation d'un module.....	16
5.4.22	Module distribué.....	16
5.4.23	Type de licences.....	17
6	Modèle de données.....	18
7	Composants techniques.....	22

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

1.2 Table des illustrations

Diagramme 1 – Extension d'un concept de base	7
Diagramme 2 – Modèle physique des données publiques du module <i>Up! 5GL</i>	20
Tableau 3 – Glossaire du modèle physique des données publiques du module <i>Up! 5GL</i>	21
Tableau 4 – Composants techniques du module	24

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

2 Références

2.1 Glossaire

Liste des définitions des termes employés.	
Ce tableau recense tous les termes, les concepts particuliers ainsi que les abréviations employés dans ce document.	
Terme, concept, abrégé.	Définition du terme, du concept ou de l'abréviation.

2.2 Ressources

Liste des documents applicables et en référence.		
Un document est applicable à partir du moment où son contenu est validé et que l'activité ou le projet fait partie de son périmètre d'application. Il est obligatoire d'appliquer son contenu.		
Un document est en référence à partir du moment où son contenu n'est pas validé ou que l'activité ou le projet ne fait partie de son périmètre d'application. Il est recommandé d'appliquer son contenu mais cela n'est pas obligatoire.		
Un document applicable est indicé par A1, A2, A3 , etc. Un document en référence est indicé par R1, R2, R3 , etc.		
Index.	Nom du document.	Commentaire.
A1	UpComp-Plan Qualité-000005	Méthode documentaire.
A2	UpComp-Plan Qualité-000006	Processus de management de projet.
A3	UpComp-Plan Qualité-000046	Méthode de spécification technique d'un module.
A4	UpComp-Ups5gl-000002	Plan documentaire du projet.
A5	UpComp-UpsVm-000003	Plan de programmation.
A6	UpComp-UpsVm-000004	Programmation en C-- .
R7	http://www.up-comp.com	Site Internet d'Up ! Application System .
A8	UpComp-UpsKrn-000003	Plan d'écriture d'un module.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

3 Introduction

3.1 Objet du document

L'objet de ce document est de décrire le contenu technique du module logiciel **Up ! 5GL** pour le projet **Up ! Application System**.

Ce document est rédigé et approuvé par la **Maîtrise d'Oeuvre (MOE)**.

3.2 Audience

Ce document s'adresse aux :

- **Directeurs de projets et chefs de projets.**
Pour la compréhension du module technique.
- **Ingénieurs de développement.**
Pour savoir comment est conçu le module technique.

Pour aider ces personnes à remplir le document **Spécification technique d'un module**, leur manager et la cellule de support projet se tiennent à leur disposition.

3.3 Pré-requis

Le pré-requis est la connaissance des documents suivants :

- **Méthode documentaire** [A1].
- **Processus de management de projet** [A2].
- **Méthode de spécification technique d'un module** [A3].

Nous rappelons que tous les documents applicables ou référencés pour le projet **Up ! Application System** sont tracés dans le **Plan documentaire** [A4].

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

4 Interaction avec l'environnement

4.1 Description

L'objet du module logiciel *Up ! 5GL* est de construire les définitions et les instructions de base du noyau du langage de cinquième génération.

4.2 Paramètres

Up ! 5GL ne possède aucun paramètre.

4.3 Particularités

4.3.1 Compilation

Néant.

4.3.2 Exécution

Néant.

4.4 Application Program Interfaces

Néant.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

5 Choix techniques

5.1 Principes

Up ! 5GL modélise les concepts de base i.e. strictement nécessaires des langages de programmation classiques :

- Constantes.
- Enumérés.
- Types.
- Variables.
- Expressions.
- Instructions.
- Sources.

Les concepts plus élaborés, voire spécifiques, sont obtenus par transformation ou extension d'un ou de plusieurs concepts de base et par compléments au travers d'objets dédiés mis en oeuvre par **Up ! Virtual Technical Machine**.

5.2 Extension du noyau linguistique

Le noyau linguistique peut être étendu via un ou plusieurs modules secondaires dont les caractéristiques sont les suivantes, dans la limite de **CO_NbLangages1Max** extensions :

- **L'interface de données doit être une extension de Lg1XDonnees.**
Ceci permet de spécifier à **Up ! 5GL** la taille de l'espace mémoire privé nécessaire pour l'extension de chaque concept de base.
Quand aucun espace mémoire n'est pas nécessaire, la taille à spécifier est **0**.
- **L'interface de traitements doit être une extension de Lg1XTraitements.**
Ceci permet de spécifier à **Up ! 5GL** les fonctions de rappels nécessaires pour étendre les algorithmes sur chaque concept de base.
Quand une fonction de rappel n'est pas nécessaire, la fonction de rappel doit être **UpsKrn.AppellImpossibleApiNull**.

Chaque concept comporte une table d'extension dénommée **TableExtension** conservant l'espace mémoire privé nécessaire à un module particulier. L'index dans cette table est le numéro d'extension du module donné par **Lg1XDonnees.EnteteOption.NumeroOption**.

Voici le schéma d'une extension d'un concept de base :

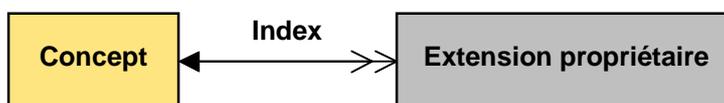


Diagramme 1 – Extension d'un concept de base

M

Les extensions possibles des concepts ne sont pas représentées dans le modèle de données.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

5.3 Analyseurs sémantiques

L'analyse sémantique doit débuter par l'appel à l'**Applications Program Interfaces (API) DebuterAnalyseSemantique** et il est détruit par l'appel à l'**Applications Program Interfaces (API) TerminerAnalyseSemantique**.

Un analyseur sémantique est identifié par un entier. Un nouvel analyseur sémantique est créé par l'appel à l'**Applications Program Interfaces (API) AjouterAnalyseurSemantique** et il est détruit par l'appel à l'**Applications Program Interfaces (API) AjouterAnalyseurSemantique**.

Un paquet sémantique est identifié par un entier. Un nouvel analyseur sémantique est créé par l'appel à l'**Applications Program Interfaces (API) AllouerPaquetSemantique** et il est détruit par l'appel à l'**Applications Program Interfaces (API) ReallouerPaquetSemantique**.

La plupart des **Applications Program Interfaces (API)** comporte deux paramètres :

- **NumeroSemantique.**
Pour identifier l'analyseur sémantique de travail.
- **NumeroPaquet.**
Pour identifier le paquet sémantique de travail.

Suite à l'analyse de l'interface des modules d'**Up ! Kernel**, d'**Up ! System** et d'**Up ! Network**, les **Applications Program Interfaces (API)** suivantes doivent être respectivement appelées :

- **RecupererObjetsNoyauUpsKrn.**
- **RecupererObjetsNoyauUpsSys.**
- **RecupererObjetsNoyauUpsNet.**

A l'issu, les analyseurs sémantiques sont réellement opérationnels.

5.4 Codifications

5.4.1 Constantes et énumérés

La nature de la valeur d'une constante ou d'un énuméré est codifiée par l'énuméré **EnuUps5GLValeurConstante** :

- **VC_Enumere.**
La valeur est une valeur énumérée.
- **VC_Entier.**
La valeur est une valeur entière.
- **VC_Reel.**
La valeur est une valeur réelle.
- **VC_Caractere.**
La valeur est une chaîne de caractères.
- **VC_PasDeValeur.**
Pour un énuméré non valué.

5.4.2 Types

La nature d'un type stricto sensu est définie par l'énuméré **EnuUps5GLType** :

- **TY_Alias.**
Pour un alias posant une contrainte sur un type, équivalent au type **Type**.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

- **TY_Fonction.**
Pour une fonction équivalente au type *Appel*.
- **TY_Interface.**
Pour une interface équivalente au type *Type*.
- **TY_Procedure.**
Pour une procédure équivalente au type *Appel*.
- **TY_Type.**
Pour un type équivalent au type *Type*.
- **TY_Extension.**
Pour un nouveau concept de type défini par à un autre module linguistique.

5.4.3 Comparaison de types

Le résultat de la comparaison de deux types est défini par l'énuméré *EnuUps5GLComparaisonType* :

- **CT_Identique.**
Les deux types sont identiques.
- **CT_Different.**
Les deux types sont différents.
- **CT_IdentiqueSaufNul.**
Les deux types sont identiques, sauf en ce qui concerne les *Nul* ou.
- **CT_IdentiqueSiParametreDefault.**
Les deux types sont deux appels de procédure, de fonction ou de méthode. Les prototypes sont identiques, si les valeurs par défaut des paramètres sont utilisées.
- **CT_IdentiqueSaufValeurDefault.**
Les deux types sont deux appels de procédure, de fonction ou de méthode. Les prototypes sont identiques, sauf si les valeurs par défaut des paramètres sont utilisées.
- **CT_DifferentSaufSiParametreDefault.**
Les deux types sont deux appels de procédure, de fonction ou de méthode. Les prototypes sont différents, sauf si les valeurs par défaut des paramètres sont utilisées.
- **CT_TropDeParametres.**
Les deux types sont deux appels de procédure, de fonction ou de méthode. Les prototypes sont différents, parce que le second a plus de paramètres de que le premier.
- **CT_PasAssezDeParametres.**
Les deux types sont deux appels de procédure, de fonction ou de méthode. Les prototypes sont différents, parce que le second a moins de paramètres de que le premier.

5.4.4 Paramètres d'un type

La nature d'un paramètre d'un type est définie par l'énuméré *EnuUps5GLParametreType* :

- **PT_Constante.**
Le paramètre du type est une constante.
- **PT_Enumere.**
Le paramètre du type est un énuméré.
- **PT_Type.**
Le paramètre du type est un type.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

5.4.5 Contraintes

La nature d'une contrainte est définie par l'énuméré *EnuUps5GLContrainte* :

- **CT_PasDeContrainte.**
Le type ne comporte pas de contrainte.
- **CT_Superieur.**
La contrainte correspond à l'opérateur supérieur strict.
- **CT_SuperieurOuEgal.**
La contrainte correspond à l'opérateur supérieur ou égal.
- **CT_Inferieur.**
La contrainte correspond à l'opérateur inférieur strict.
- **CT_InferieurOuEgal.**
La contrainte correspond à l'opérateur inférieur ou égal.
- **CT_Entre.**
La contrainte correspond à l'opérateur **Entre**.
- **CT_Dans.**
La contrainte correspond à l'opérateur **Dans**.
- **CT_Comme.**
La contrainte correspond à l'opérateur **Comme**.
- **CT_Appel.**
La contrainte correspond à une méthode spéciale.

5.4.6 Méthodes

La nature d'une méthode est définie par l'énuméré *EnuUps5GLInflexionMethode* :

- **IM_Normal.**
La méthode ne porte aucune sémantique particulière.
- **IM_Constructeur.**
La méthode est le constructeur implicite.
- **IM_Destructeur.**
La méthode est le destructeur implicite.
- **IM_Allouer.**
La méthode est l'allocateur d'objets.
- **IM_Liberer.**
La méthode est le libérateur d'objets.
- **IM_AllouerRessource.**
La méthode est l'allocateur de ressource.
- **IM_LibererRessource.**
La méthode est le libérateur de ressource.
- **IM_Implicite.**
La méthode est implicite i.e. automatiquement définie par *Up! 5GL*.

5.4.7 Résultat d'un prototype

La nature du résultat d'un prototype est définie par l'énuméré *EnuUps5GLPrototype* :

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

- **PR_Fonction.**
Le prototype est obligatoirement celui d'une fonction.
- **PR_Procedure.**
Le prototype est obligatoirement celui d'une procédure.
- **PR_FonctionOuProcedure.**
Le prototype est soit celui d'une procédure ou soit celui d'une fonction.

5.4.8 Récurtivité d'un type

La récurtivité d'un type ou d'un prototype d'un appel de procédure ou de fonction est définie par l'énuméré **EnuUps5GLRecurtivité** :

- **RE_NonRecurtif.**
Le type, la procédure ou la fonction n'est pas récurtif.
- **RE_RecurtifDefini.**
Le type, la procédure ou la fonction est récurtif. D'autre part, la définition est posée.
- **RE_RecurtifNonDefini.**
Le type, la procédure ou la fonction est récurtif. D'autre part, la définition n'est pas posée.

5.4.9 Composition des opérateurs

La composition des opérateurs est définie par l'énuméré **EnuUps5GLCompositionOperateur** :

- **CO_Direct.**
La composition n'est pas nécessaire. Le premier opérateur est appliqué directement.
- **CO_Enchainer.**
La composition est obtenue par application en séquence des deux opérateurs.
- **CO_Et.**
La composition est obtenue par la conjonction inclusive du résultat des deux opérateurs.
- **CO_EtNon.**
La composition est obtenue par la conjonction inclusive du résultat du premier opérateur et de la négation du résultat du second opérateur.
- **CO_Inconnu.**
Il n'y a pas de composition des opérateurs possible.
- **CO_Non.**
La composition est obtenue par la négation du résultat de l'opérateur.
- **CO_NonEt.**
La composition est obtenue par la négation de la conjonction inclusive du résultat des deux opérateurs.
- **CO_NonOu.**
La composition est obtenue par la négation de la disjonction inclusive du résultat des deux opérateurs.
- **CO_Ou.**
La composition est obtenue par la disjonction inclusive du résultat des deux opérateurs.
- **CO_OuNon.**
La composition est obtenue par la disjonction inclusive du résultat du premier opérateur et de la négation du résultat du second opérateur.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

5.4.10 Factorisation des opérateurs booléens

La factorisation des opérateurs est définie par l'énuméré *EnuUps5GLFactorisationBooleen* :

- **FC_Incomparable.**
Les opérateurs ne sont pas factorisables.
- **FC_Vrai.**
Le résultat est toujours *Vrai*.
- **FC_Faux.**
Le résultat est toujours *Faux*.
- **FC_Factorisable.**
Les opérateurs sont factorisables.

5.4.11 Expressions

La nature d'une expression est définie par l'énuméré *EnuUps5GLExpression* :

- **EX_ChaineGuillemets.**
L'expression est une chaîne de caractères entre caractères **guillemet** " .
- **EX_Champ.**
L'expression est la sélection d'une propriété d'un type ou d'une interface.
- **EX_Constante.**
L'expression est une constante.
- **EX_Constructeur.**
L'expression est le foncteur d'un constructeur d'un type ou d'une interface.
- **EX_Conversion.**
L'expression est une conversion d'un type ou d'une interface dans un autre.
- **EX_Entier.**
L'expression est une constante entière.
- **EX_Entrepot.**
L'expression est un entrepôt.
- **EX_Enumere.**
L'expression est une énuméré.
- **EX_Evaluation.**
L'expression l'évaluation d'un appel de procédure, de fonction ou d'une méthode.
- **EX_Fonction.**
L'expression est le foncteur d'une procédure ou d'une fonction.
- **EX_Joker.**
L'expression est l'opérateur ?.
- **EX_Methode.**
L'expression est la sélection d'une méthode d'un type ou d'une interface.
- **EX_Module.**
L'expression est un module.
- **EX_Nul.**
L'expression est *Nu1*.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

- **EX_Objct.**
L'expression est **Objct**.
- **EX_OperateurBinaire.**
L'expression est un opérateur binaire appliqué à deux expressions.
- **EX_OperateurUnairePostfixe.**
L'expression est un opérateur unaire post-fixé appliqué à une expression.
- **EX_OperateurUnairePrefixe.**
L'expression est un opérateur unaire préfixé appliqué à une expression.
- **EX_Parenthese.**
L'expression est la mise entre parenthèses d'une expression.
- **EX_Reel.**
L'expression est une constante réelle.
- **EX_Super.**
L'expression est le sélecteur d'une super méthode.
- **EX_Type.**
L'expression est un type ou une interface.
- **EX_ValeurEnumere.**
L'expression est une valeur d'un énuméré.
- **EX_Variable.**
L'expression est une variable locale ou globale.
- **EX_Extension.**
Pour un nouveau concept d'expression défini par à un autre module linguistique.

5.4.12 Opérateurs binaires

La nature d'un opérateur binaire est définie par l'énuméré **EnuUps5GLSorteOperateur** :

- **SO_Comparaison.**
Opérateur binaire tel **==**.
- **SO_Booleen.**
Opérateur binaire tel **Et**.
- **SO_Autre.**
Autre binaire tel **Comme**.

5.4.13 Comparaison d'expressions

Le résultat de la comparaison de deux expressions est défini par l'énuméré **EnuUps5GLComparaisonExpression** :

- **CE_Identique.**
Les deux expressions sont identiques.
- **CE_Different.**
Les deux expressions sont différentes.
- **CE_GaucheContientDroite.**
L'expression de gauche contient l'expression de droite.
- **CE_DroiteContientGauche.**
L'expression de droite contient l'expression de gauche.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

- **CE_Incomparable.**
Lors de la preuve de programme, pour estimer si la seconde expression est constante, sachant que la première est établie, la seconde expression n'est pas comparable à la première.
- **CE_Vrai.**
Lors de la preuve de programme, pour estimer si la seconde expression est constante, sachant que la première est établie, la seconde expression est toujours évaluée à **Vrai**.
- **CT_Faux.**
Lors de la preuve de programme, pour estimer si la seconde expression est constante, sachant que la première est établie, la seconde expression est toujours évaluée à **Faux**.
- **CE_VraiEnPartie.**
Lors de la preuve de programme, pour estimer si la seconde expression est constante, sachant que la première est établie, la seconde expression est toujours évaluée à **Vrai**, sauf dans le cas aux limites

5.4.14 Héritages

Le mode d'héritage d'une définition est défini par l'énuméré **EnuUps5GLHeritage** :

- **HE_Public.**
L'héritage de la définition est public.
- **HE_Protege.**
L'héritage de la définition est protégé.
- **HE_Prive.**
L'héritage de la définition est privé.

5.4.15 Passages de paramètres

Le mode de passage des paramètres est défini par l'énuméré **EnuUps5GLPassageParametre** :

- **PA_Entree.**
Le paramètre est passé en entrée.
- **PA_Sortie.**
Le paramètre est passé en sortie.
- **PA_EntreeSortie.**
Le paramètre est passé en entrée et en sortie.

5.4.16 Valeurs par défaut des paramètres

La nature de la valeur par défaut des paramètres est définie par l'énuméré **EnuUps5GLValeurDefautParametre** :

- **DP_Obligatoire.**
Le paramètre est obligatoire.
- **DP_Nul.**
Le paramètre a pour valeur par défaut **Nul**.
- **DP_Constante.**
Le paramètre a pour valeur par défaut une constante.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

5.4.17 Variables

La sorte d'une variable est définie par l'énuméré *EnuUps5GLPrototype* :

- **SV_Normale.**
La variable est soit locale ou soit globale.
- **SV_ParametreAppel.**
La variable est un paramètre d'une procédure, d'une fonction ou d'une méthode d'un type ou d'une interface.
- **SV_ParametreComposant.**
La variable est un paramètre d'un composant du module.

5.4.18 Instructions

La nature d'une instruction est définie par l'énuméré *EnuUps5GLInstruction* :

- **IN_Expression.**
Instruction correspondant à une expression telle une affectation ou un appel de procédure, de fonction ou de méthode.
- **IN_Arreter.**
Instruction **Arreter**.
- **IN_Continuer.**
Instruction **Continuer**.
- **IN_Retourner.**
Instruction **Retourner**.
- **IN_Pour.**
Instruction **Pour**.
- **IN_TantQue.**
Instruction **TantQue**.
- **IN_Faire.**
Instruction **Faire**.
- **IN_Si.**
Instruction **Si**.
- **IN_SinonSi.**
Instruction **SinonSi**.
- **IN_Sinon.**
Instruction **Sinon**.
- **IN_Extension.**
Pour un nouveau concept d'instruction défini par à un autre module linguistique.

5.4.19 Objets

La nature d'un objet est définie par l'énuméré *EnuUps5GLObjet* :

- **OB_Appel.**
L'objet est un foncteur de procédure, de fonction ou de méthode d'un type ou d'une interface.
- **OB_Champ.**
L'objet est une propriété d'un objet.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

- **OB_Constante.**
L'objet est une constante.
- **OB_Enumere.**
L'objet est un énuméré.
- **OB_Methode.**
L'objet est une méthode d'un objet.
- **OB_Type.**
L'objet est un type.
- **OB_ValeurEnumere.**
L'objet est une valeur d'un énuméré.
- **OB_Variable.**
L'objet est une variable.
- **OB_Extension.**
Pour un nouveau concept d'objet défini par à un autre module linguistique.

5.4.20 Types de fichiers sources

Le type de fichiers sources est défini par l'énuméré *EnuUps5GLTypeSource* :

- **TS_SourceComposant.**
Le fichier source est celui d'un composant.
- **TS_SourceModule.**
Le fichier source est celui d'un module.
- **TS_InterfaceComposant.**
Le fichier source est celui de l'interface d'un composant.
- **TS_InterfaceModule.**
Le fichier source est celui de l'interface d'un module.
- **TS_Extension.**
Pour un nouveau concept de fichier source défini par à un autre module linguistique.

5.4.21 Implémentation d'un module

Le mode d'implémentation d'un module est défini par l'énuméré *EnuUps5GLImplementationModule* :

- **IM_Statique.**
Le module est statique.
- **IM_Dynamique.**
Le module est dynamique.

5.4.22 Module distribué

Le mode de distribution d'un module est défini par l'énuméré *EnuUps5GLTypeDistribution* :

- **TD_Corba.**
La liaison s'effectue via *Common Object Request Broker Architecture* (CORBA) de l'*Object Management Group* (OMG).
- **TD_DCom.**
La liaison s'effectue via *Distributed Component Object Module* (DCOM) de *Microsoft*.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

- **TD_Java.**
La liaison s'effectue via **Java**.
- **TD_UpsNet.**
La liaison s'effectue via **Up ! Network**.

5.4.23 Type de licences

Le type de licence d'un module est défini par l'énuméré **EnuUps5GLTypeLicence** :

- **TL_Developpement.**
Il s'agit d'une licence de développement.
- **TL_Personnalisation.**
Il s'agit d'une licence de personnalisation.
- **TL_ExecutionInteractive.**
Il s'agit d'une licence d'exécution en interactif.
- **TL_ExecutionBatch.**
Il s'agit d'une licence d'exécution en batch.



Spécification technique du module

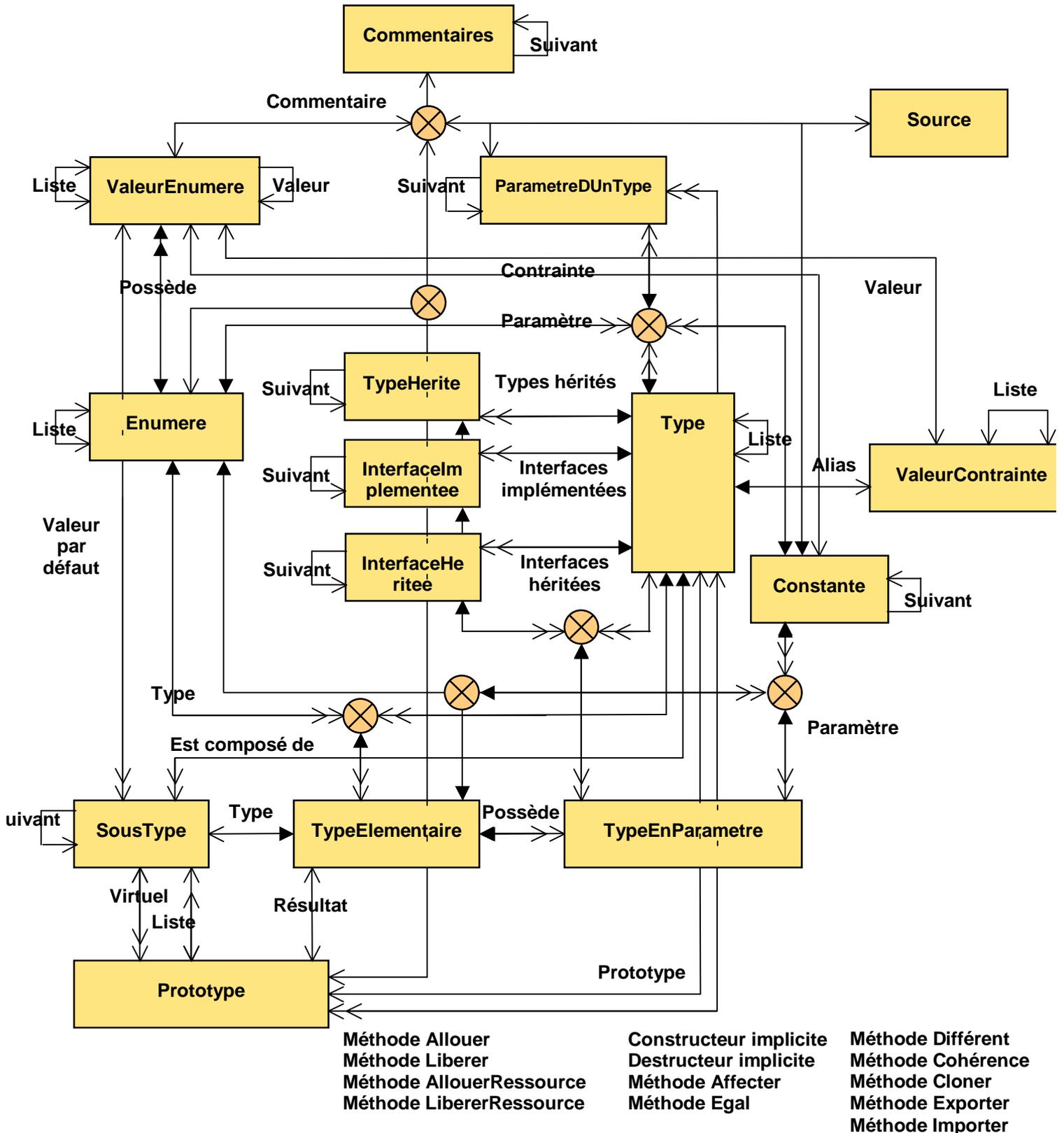
Date rédaction :
14 mars 2004.

Diffusion restreinte

Date validation :

Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc

6 Modèle de données



	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

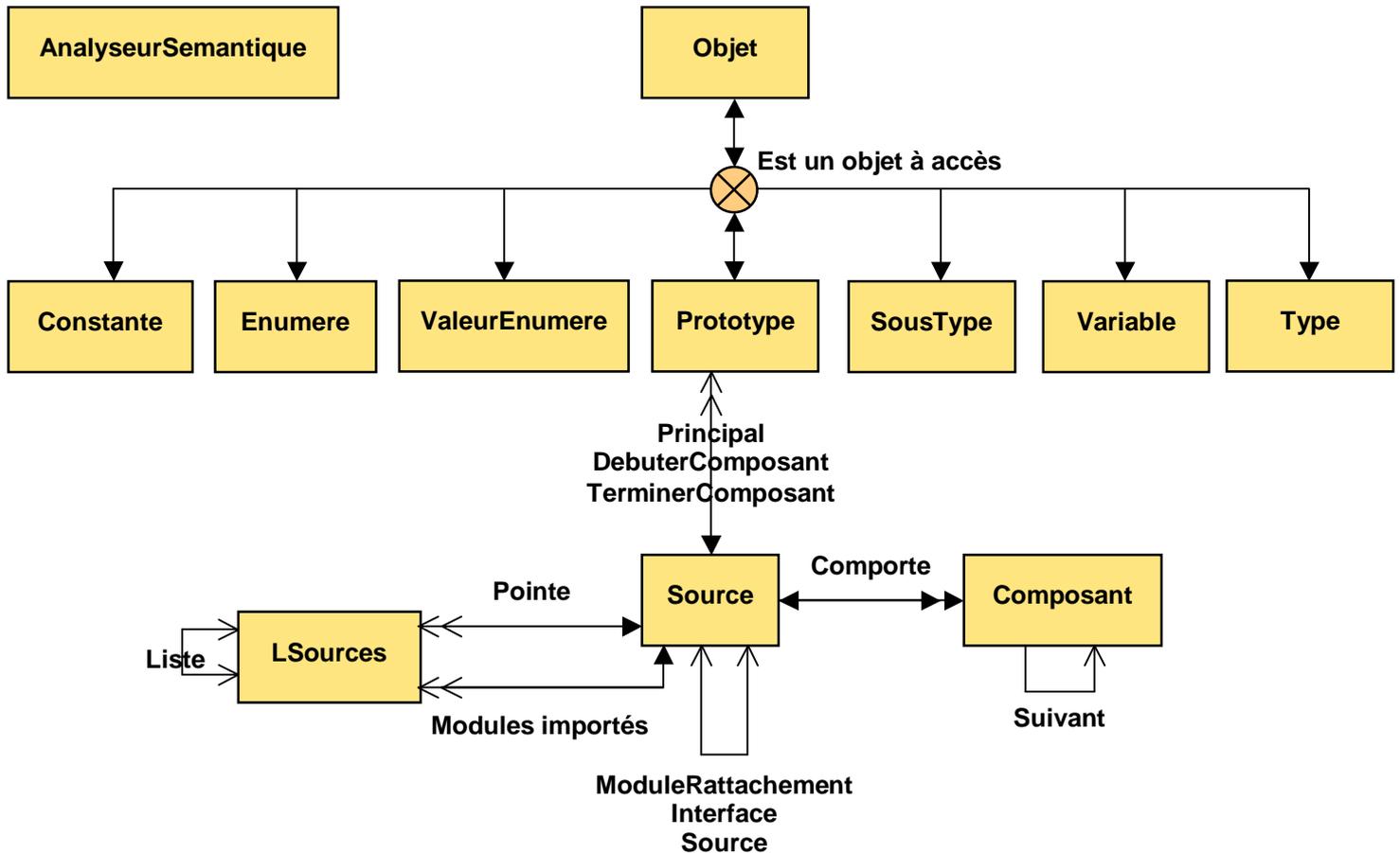


Diagramme 2 – Modèle physique des données publiques du module Up ! 5GL

M Le modèle de données utilise des allocateurs et des tables, à des fins d'optimisation, qui ne sont pas représentés sur le diagramme.

Toutes les entités suivantes sont décrites dans le fichier **usp5gl.e** :

Entité.	Description.
AnalyseurSemantique.	Identifiant d'un analyseur sémantique.
Appel.	Foncteur d'un appel de procédure, de fonction ou de méthode.
Commentaires.	Commentaires sur une définition.
Composant.	Composant d'un module.
Constante.	Constante dont la valeur est énumérée, entière, réelle ou chaîne de caractères.
Entrepot.	Entrepôt de création des objets.
Enumere.	Enuméré comportant au moins une valeur énumérée.
Expression.	Expression élémentaire.
Instruction.	Instruction élémentaire.
InterfaceHeritee.	Interface héritée par une autre interface.

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

InterfaceImplémentee. LAppels. LSources. Objet. PaquetSemantique. ParametreDUnType. Prototype. Source. SousType. Type. TypeElementaire. TypeEnParametre. TypeHerite. ValeurContrainte. ValeurEnumere. Variable.	Interface implémentée par un autre type. Liste d'appels pointés. Liste de fichiers sources pointés. Objet comportant un mode d'accès. Paquet de définitions locales portant une sémantique. Paramètre attendu d'un type. Prototype d'un appel de procédure, de fonction ou de méthode. Fichier source. Paramètre d'un prototype ou propriété d'un type ou d'une interface. Type ou interface. Type utilisé dans une déclaration. Paramètre effectif d'un type. Type hérité par un autre type. Contrainte élémentaire sur un type. Valeur d'un énuméré. Variable locale ou globale.
--	---

Tableau 3 – Glossaire du modèle physique des données publiques du module *Up ! 5GL*

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

7 Composants techniques

Le module *Up ! 5GL* pour le projet *Up ! Application System* est constitué des composants suivants :

Fichiers du module.	
<ul style="list-style-type: none"> Fichier ups5gl.e – Définition de <i>Up ! 5GL</i>. Fichier ups5gl.h – En-tête privé de <i>Up ! 5GL</i>. Fichier ups5gl.def – Exportation des symboles pour <i>Windows</i>. 	
Composants.	ups5gl0.
Description.	
Interface entre <i>Up ! 5GL</i> et <i>Up ! Module</i> .	
Fichiers.	
<ul style="list-style-type: none"> Fichier ups5gl0.cpp – Fichier source. Fichier ups5gl0.h – En-tête privé de ups5gl0.cpp. Fichier ups5gl0.e – En-tête protégé de ups5gl0.cpp. 	
Composants.	ups5gl1.
Description.	
Utilitaires pour l'analyseur sémantique générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier ups5gl1.cpp – Fichier source. Fichier ups5gl1.h – En-tête privé de ups5gl1.cpp. Fichier ups5gl1.e – En-tête protégé de ups5gl1.cpp. 	
Composants.	ups5gl2.
Description.	
Allocateurs et libérateurs pour l'analyseur sémantique générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier ups5gl2.cpp – Fichier source. Fichier ups5gl2.h – En-tête privé de ups5gl2.cpp. Fichier ups5gl2.e – En-tête protégé de ups5gl2.cpp. 	
Composants.	ups5gl3.
Description.	
Gestion des types pour l'analyseur sémantique générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier ups5gl3.cpp – Fichier source. Fichier ups5gl3.h – En-tête privé de ups5gl3.cpp. Fichier ups5gl3.e – En-tête protégé de ups5gl3.cpp. 	
Composants.	ups5gl4.
Description.	

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		

Services de cohérence pour l'analyseur sémantique générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier ups5gl4.cpp – Fichier source. Fichier ups5gl4.h – En-tête privé de ups5gl4.cpp. Fichier ups5gl4.e – En-tête protégé de ups5gl4.cpp. 	
Composants.	ups5gl5.
Description.	
Gestion des appels pour l'analyseur sémantique générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier ups5gl5.cpp – Fichier source. Fichier ups5gl5.h – En-tête privé de ups5gl5.cpp. Fichier ups5gl5.e – En-tête protégé de ups5gl5.cpp. 	
Composants.	ups5gl6.
Description.	
Gestion des modules pour l'analyseur sémantique générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier ups5gl6.cpp – Fichier source. Fichier ups5gl6.h – En-tête privé de ups5gl6.cpp. Fichier ups5gl6.e – En-tête protégé de ups5gl6.cpp. 	
Composants.	ups5gl7.
Description.	
Gestion des constantes et des énumérés pour l'analyseur sémantique générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier ups5gl7.cpp – Fichier source. Fichier ups5gl7.h – En-tête privé de ups5gl7.cpp. Fichier ups5gl7.e – En-tête protégé de ups5gl7.cpp. 	
Composants.	ups5gl8.
Description.	
Construction des types complexes pour l'analyseur sémantique générique.	
Fichiers.	
<ul style="list-style-type: none"> Fichier ups5gl8.cpp – Fichier source. Fichier ups5gl8.h – En-tête privé de ups5gl8.cpp. Fichier ups5gl8.e – En-tête protégé de ups5gl8.cpp. 	
Composants.	ups5gl99.
Description.	
Interface entre Up ! 5GL et Up ! Kernel .	
Fichiers.	

	Spécification technique du module	Date rédaction : 14 mars 2004.
	Diffusion restreinte	Date validation :
Référence : UpComp-Ups5gl-000003-A Spécification technique du module Ups5gl.doc		
<ul style="list-style-type: none"> • Fichier ups5gl99.cpp – Fichier source. • Fichier ups5gl99.h – En-tête privé de ups5gl99.cpp. • Fichier ups5gl99.e – En-tête protégé de ups5gl99.cpp. 		

Tableau 4 – Composants techniques du module

Fin de document