Direction de la recherche et de la technologie
Unité de recherche « Génie Décisionnel Appliqué »
45 Rue de Londres
75379 Paris Cedex 08

RAPPORT DE STAGE

SNCF

Génération automatique de diagrammes signaux/ vitesses sous AutoCAD

Stage effectué du 10 juin 2003 au 30 septembre 2003

Tuteur : Mr Castanet		Qualité : Enseignant Chercheur
Maître de stage : Mme Chantal Joie		Qualité : Chef de projet
Alaoui Mohamed	Département : Informatique	15 septembre 2003

TABLE DES MATIERES

REMERCIEMENTS....3

PARTIE N°1. INTRODUCTION....4

PARTIE N°2. L'ENTREPRISE SNCF....5

Raison sociale, statut juridique et actionnariat....5
Chiffres clefs et Activités....6
Concurrence....7
Recherche et développement....7
Organisation....8
Les métiers de l'ingénieur au sein de l'ENTREPRISE SNCF....8

PARTIE N°3. OBJECTIFS DE LA MISSION TECHNIQUE....10

Sujet de stage....10
Contexte du sujet....11
Cahier des charges fonctionnel....14
Critères de validation du projet....19
Moyens mis à disposition du stagiaire....19
Planification du projet....19

PARTIE N°4. RÉALISATION DU PROJET....20

Description de l'existant....20
Analyse du cahier des charges et de ses contraintes....27
Conception générale....29
Conception détaillée....30
Résultats....47

PARTIE N°5. CONCLUSION....51

PARTIE N°6. GLOSSAIRE....52

PARTIE N°7. BIBLIOGRAPHIE....53

Remerciements

Mes remerciements s'adressent à l'ensemble du personnel du service de la Direction de la Recherche et de la Technologie qui m'a très bien accueilli et encadré. En particulier Mme Chantal Joie, mon maître de stage, Mr Jean-Luc Vassent, le prestataire responsable du projet englobant mon sujet, pour son encadrement, son aide et ses explications très précieuses et enfin tous les stagiaires du service GDA.

Je tiens à remercier également Mme Faucher du département de langue de l'ENSEIRB qui m'a aidé lors de la recherche de stages ainsi que Melle von Ungern Sternberg pour son soutien.

PARTIE N°1.INTRODUCTION

Mon objectif de stage de 2ème année était de participer à un projet, en entreprise, à valeur ajoutée technologique utilisant en particulier le concept objet. Le sujet proposé par l'unité GDA (génie décisionnel appliqué) de la Direction de la Recherche et de la Technologie de la SNCF correspondait exactement à mes attentes.

La finalité de mon stage est de réaliser un prototype et son Interface Homme Machine permettant de dessiner automatiquement des diagrammes signaux/vitesses représentant l'implantation des signaux et des zones de limitation de vitesse pour un itinéraire de train donné. Ces diagrammes constituent les documents de bases pour le dépouillement des « boîtes noires » des trains. Ces documents sont jusqu'à présent réalisés à la main sous une application de dessin.

Dans le cadre d'un projet plus global, né d'un besoin exprimé par la Direction de la Traction de la SNCF visant à automatiser l'exploitation des données de suivi de conduite sur les trains équipés du système ATESS (« boîte noire » informatisée), un modèle conceptuel de données a été élaboré permettant de modéliser les diagrammes signaux/vitesses. La base de données permettant de décrire un DSV existe donc déjà, ainsi que le mapping Objet/Relationnel qui fait l'interface entre les objets métiers (BusinnesObjects) et la base de données.

Ce stage consiste à utiliser ce mapping ainsi que l'API ObjectARX d'AutoCAD pour tracer automatiquement ces diagrammes sous AutoCAD 2002.

Le développement est effectué en C++ sous Visual C++ 6.0 sur Windows NT. La base de données utilisée est Access 97. AutoCAD 2002 est utilisée pour la génération du dessin. L'IHM est réalisée avec la bibliothèque MFC de Microsoft Windows.

PARTIE N°2.L'ENTREPRISE SNCF

Raison sociale, statut juridique et actionnariat

La SNCF est devenue un Etablissement Public Industriel et Commercial (EPIC) régi par les articles 18 à 26 de la Loi d'Orientation des Transports Intérieurs du 30 décembre 1982 (LOTI) avec effet au 1er janvier 1983, après un statut de Société Anonyme d'économie mixte pendant 45 années, prenant effet le 1er janvier 1938 (Convention du 31 août 1937).

Depuis la modification apportée par la loi du 13 février 1997 portant sur la création de l'établissement public "Réseau Ferré de France" (RFF) en vue du renouveau du transport ferroviaire, la SNCF a pour objet l'exploitation, selon les principes du service public, des services de transport ferroviaire sur le réseau ferré national, l'assurance, selon les mêmes principes, des missions de gestion de l'infrastructure par un mandat qui lui est confié par Réseau ferré de France sous forme de convention.

La SNCF est dotée d'un conseil d'administration composé de 18 membres dont sept représentants de l'Etat, cinq membres choisis en raison de leur compétence, dont au moins un représentant des usagers nommés par décret et enfin six membres, dont un représentant des cadres, élus par les salariés de l'entreprise et de ses filiales.

Le Président du Conseil d'administration est nommé, sur proposition de celui-ci, par décret au Conseil des ministres.

La SNCF est pilotée par un Comité Exécutif composé de :

Louis GALLOIS, Président .
Guillaume PEPY, Directeur Général Délégué Clientèles.
Jacques COUVERT, Directeur Général Délégué Exploitation.
Bernard EMSELLEM, Directeur de la Communication.
Pierre IZARD, Directeur des Ressources Humaines.
Claire DREYFUS-CLOAREC, Directeur Economie – Finances.
Paul MINGASSON, Secrétaire Général.

Les Directions et Services Centraux sont rattachés au Comité Exécutif.
Les 23 Directeurs de Régions sont placés directement sous l'autorité du Président.
La SNCF est dotée de l'autonomie de gestion. Un cahier des charges, approuvé par décret, fixe ses droits et obligations. C'est en application de ce document que sont notamment déterminés les

différents concours financiers de l'Etat à l'entreprise (contribution aux charges de retraites, tarifs sociaux, …). Le cahier des charges fixe également les règles qui président à l'élaboration des tarifs tant voyageurs que fret. A cet égard, seuls les tarifs voyageurs font l'objet d'une homologation du ministre chargé des transports, les tarifs marchandises étant simplement communiqués.

En tant qu'entreprise publique, elle est soumise au contrôle à posteriori de la Cour des Comptes. En outre, l'Etat a organisé un contrôle économique et financier, à priori, qui est exercé par la Mission de contrôle économique et financier des chemins de fer, organe permanent, installé au siège de l'entreprise. Elle a pour mission de formuler un avis écrit sur toutes les questions soumises au Conseil d'administration ainsi que sur toutes questions et tous projets de décision ayant une incidence sur l'équilibre financier de la SNCF.

La SNCF compte actuellement 181 565 personnes (31.12.2001) réparties sur environ 305 établissements.

Chiffres clefs et Activités

Pourkoi pas de petits tableau un peu plus clair avec deux trosi phrases explicatives a la fin? Ca pourrait faire un jolie lien à la concurrence…

- **RÉSULTAT DU GROUPE SNCF (en millions d'euros)**

	2001	**2002**
Chiffre d'affaires hors taxes	20 129	22 176
Excédent brut d'exploitation	1 155	1 403
Résultat d'exploitation	20	183
Résultat courant des sociétés intégrées	- 310	- 137
Résultat net part du Groupe	- 140	63

- **RÉSULTATS PAR BRANCHE D'ACTIVITÉ**

	Transport de voyageurs		*Transport de marchandises*		*Infrastructure, valorisation du patrimoine et du savoir-faire*	
	2001	*2002*	*2001*	*2002*	*2001*	*2002*
Chiffre d'affaires	8 562	10 864	6 622	6 345	4 945	4 967
Excédent brut d'exploitation	888	1209	37	- 87	230	281
Résultat d'exploitation	230	450	- 283	– 360	73	94

- **ÉTABLISSEMENT PUBLIC**

Comptes annuels	2001	2002
Chiffre d'affaires hors taxes	14 227	14 782
Excédent brut d'exploitation	664	713
Résultat d'exploitation	- 71	- 17
Résultat courant	- 176	/ - 183
Résultat net	- 134	19
Investissements hors taxes	1 330	1 725
Effectifs moyens payés (hors emplois jeunes)	??	18
Frais de personnel	7 493	7 653

Concurrence

Louis Gallois, président de la SNCF, a réaffirmé dernièrement que l'entreprise ferroviaire n'avait pas d'autre choix que de se préparer à la concurrence, ce qui lui permettrait de dominer un marché européen libéralisé.

Rappelant les échéances de libéralisation - le marché du transport de fret international sera complètement ouvert le 1er janvier 2006 - le président de la SNCF a appelé les cheminots à avoir une position pragmatique face au "principe de réalité" : "La SNCF ne peut être absente de l'Europe. L'Europe, c'est la concurrence. La SNCF doit donc être dans la concurrence", a-t-il souligné.

Pour rendre la SNCF compétitive sur le marché du transport européen, Louis Gallois a identifié trois axes de travail : l'amélioration de la compétitivité sur les coûts, le renforcement de la présence de l'entreprise à l'étranger et la définition d'une bonne politique d'investissements.

Sur la présence de la SNCF à l'étranger, Louis Gallois a fait remarquer que 20% du trafic grandes lignes était généré par l'international (Eurostar, Thalys, Artesia), soulignant que la SNCF était le "réseau le plus ouvert d'Europe".

Par ailleurs la SNCF doit faire face à une concurrence commerciale qui s'exacerbe, du fait du transport aérien.

Recherche et développement

La SNCF progresse selon deux démarches symétriques et complémentaires : d'une part l'observation du développement des technologies qui peuvent aider l'entreprise à améliorer la qualité de sa production et à créer de nouveaux produits et services et d'autre part une réflexion prospective qui prend en compte les évolutions de la clientèle et de l'environnement en terme de marchés et de concurrence.

Elle vise principalement à optimiser :

- L'exploitation du réseau ferroviaire (capacité du réseau, sécurité et régularité des circulations, compétitivité du service de transport, maintenance avancée et prédictive des infrastructures…).
- Le confort des voyageurs en gare et dans les trains (nouveaux services, nouveaux matériels ferroviaires, trains "communicants"…).
- La fiabilité et l'efficacité du fret (trains longs, localisation par GPS…).
- L'impact environnemental du transport ferroviaire (réduction des nuisances sonores, énergies nouvelles, recyclage des matériaux…).
- Les connaissances fondamentales (voie et ses composants, dynamique ferroviaire, aérodynamique, contact roue/rail…).
- L'efficacité individuelle et collective des cheminots (outils de formation par simulation, réalité virtuelle…).

Organisée en réseau et multidisciplinaire, elle s'appuie principalement sur environ 300 experts répartis entre la Direction de la Recherche et de la Technologie, où j'effectue ma mission de stage, et d'autres Directions. Elle couvre l'ensemble des domaines techniques ainsi que les sciences humaines et sociales.

Ouverte sur l'extérieur, elle travaille en partenariat avec le monde scientifique et universitaire, l'industrie ferroviaire et d'autres secteurs industriels. Elle coopère également avec des partenaires ferroviaires en Europe et dans le monde entier.

La Direction de la Recherche et de la Technologie de la SNCF publie chaque trimestre une revue, "Rail & Recherche", qui a pour vocation de rendre compte des avancées dans le domaine spécifique de la recherche ferroviaire en France et dans le monde.

Organisation

L'Organigramme complet de la SNCF ne peut être présenté ici dans sa totalité. Je vais donc présenter l'organisation de la D.R.T.

La Direction de la Recherche et de la Technologie est en charge, comme cité précédemment, de piloter le réseau de la Recherche et du Développement de l'entreprise.

On compte trois principaux groupes :

- Les unités de recherche : elles représentent les compétences scientifiques et techniques.
- Les groupes fonctionnels : ils apportent les supports de gestion et d'animation nécessaires au fonctionnement de la Recherche.
- Les groupes d'appui.

DIRECTION DE LA RECHERCHE ET DE LA TECHNOLOGIE

GROUPE D'APPUI	UNITÉS DE RECHERCHE	GROUPES FONCTIONNELS
R.H. Ressources Humaines	**S.F.C.** physique du Système Ferroviaire et Confort	**PR.D.** Prospective et Développement
C.G.L. Comptabilité – Gestion - Logistique	**G.D.A.** Génie Décisionnel Appliqué	**R.I.I.** Relations Institutionnelles et Internationales
COM. Communication	**A.S.C.** Automatismes et Systèmes de Contrôle	**INO.** Innovation
QUA. Qualité et méthodes	**N.T.I.C.** Nouvelles Technologies de l'Information, de la Communication	**V.T.P.** Veille Technologique Propriété intellectuelle
	S.H.S. Sciences Humaines et Sociales	**P.R.E.** Programme de Recherche

Les métiers de l'ingénieur au sein de l'ENTREPRISE SNCF

Lors de ce stage j'ai eu l'occasion de travailler avec deux profiles d'ingénieur différents. Mon maître de stage, Mme Chantal Joie, qui est ingénieur chef de projet et Mrs Jean Luc Vassent qui est un prestataire en informatique salarié d'une SSII.

Ingénieur Chef de Projet

Le Chef de projet SNCF est chargé de mener à terme les différents projets qui sont sous sa responsabilité. Il constitue un ensemble destiné à répondre à un cahier des charges précis dans un temps limité, en fédérant les ressources (humaines, financières, matérielles et logicielles) nécessaires, suivant un processus cohérent. Il doit rendre compte au directeur de l'unité de recherche dont il fait partie. Il a par ailleurs, sous sa responsabilité un certain nombre de prestataires et de stagiaires qui doivent lui rendre compte.

Il est chargé, en outre, avec l'aide des ingénieurs sous sa responsabilité, de définir les cahiers de charges et surtout de les faire valider par les client.

Ingénieur Prestataire en Informatique

L'ingénieur prestataire est en mission à la SNCF mais reste salarié d'une Société de Service. Il est chargé des différentes phases de développement informatique d'un projet. Il est par ailleurs chargé d'apporter toutes les solutions techniques liées au projet sur lequel il travail. Il doit rendre compte au Chef de Projet qui le supervise lors de réunion de mise au point. Il encadre par ailleurs les stagiaires travaillant sur le même projet.

PARTIE N°3.OBJECTIFS DE LA MISSION TECHNIQUE

Sujet de stage

Mon sujet de stage entre dans le cadre d'un projet plus global intitulé : **Aide à l'exploitation des données ATESS[1].**

Les besoins à satisfaire sont les suivant :
Le dépouillement détaillé des « boites noires » des trains est aujourd'hui effectué manuellement. Cette technique est inadaptée au dépouillement des données issues du système de suivi numérique ATESS mis en place sur l'ensemble des TGV et progressivement sur le reste de la flotte.
Les besoins de la Direction de la Traction , chargée entre autre du contrôle du traffic ferrovière, consistent en :

- L'élaboration d'une base de données de référence permettant le traitement automatique des données ATESS.
- La mise au point d'un moyen d'association automatique des données ATESS avec des données de référence.
- Identification et implémentation de règles de contrôle permettant un dépouillement automatique.

Les résultats attendus sont:

- Identification de la technique mathématique et/ou statistique la plus adaptée à l'association des relevés ATESS avec les données de référence (emplacement des signaux, des limitations de vitesse, des points remarquables, etc.) : réseaux de neurones, algorithmes génétiques, matheuristique, logique floue, etc.
- Construction d'une base de données de référence pour l'exploitation des données ATESS à partir des fichiers utilisés pour l'édition des bandes types[2].
- Elaboration de prototypes de traitement automatique des données ATESS.
- Identification et modélisation de mécanismes de contrôle des données de conduite.
- Implémentation de ces contrôles sur un prototype.

Dans ce cadre général, mon sujet de stage consiste à réaliser une application permettant le tracer des DSV permettant ce dépouillement. Cette application est importante car même s'il est possible d'automatiser totalement le dépouiellement, il sera toujours nécessaire de tracer les DSV sur un support papier. En effet, le dépouillement manuel restera en vigueur tant que tous les trains ne

1 « Boites Noires » informatisées.

2 Les bandes types sont des schémas présentant la liste des signaux, des aiguilles, des établissements ou autres points remarquables rencontrés par un train lors de sa progression.

seront pas équipés du système ATESS. D'autre part, la sortie papier permettra la validation des DSV.

Contexte du sujet

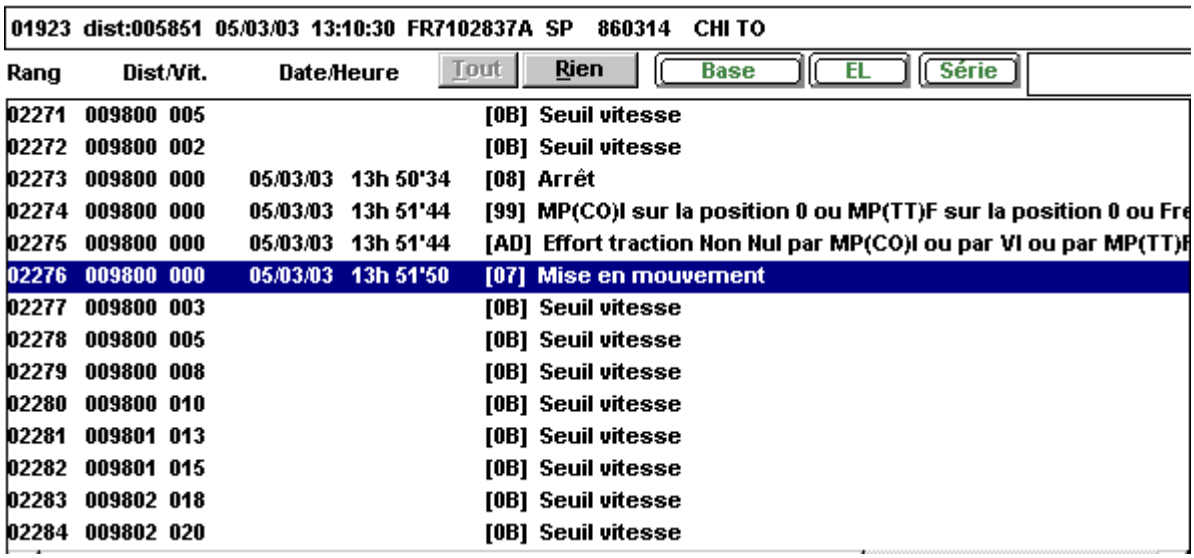
Les DSV (Diagrammes Signaux/Vitesse) sont des documents de référence qui permettent aux vérificateurs[3] d'effectuer le dépouillement des fichiers enregistrés sur les trains grâce au système ATESS, dit fichier lpb[4](Long Parcours Binaire).

Les informations concernant un DSV sont issues d'une part des bandes types (BT) élaborées par les services de l'Infrastructure à partir des plans de voies, pour les signaux, les limitations de vitesse indépendantes de la catégorie de train, les aiguilles, les gares, et d'autre part des Renseignements Techniques (RT) pour les informations de vitesse limite qui dépendent de la catégorie du train.

Fichiers ATESS et importance de la vérification

Les fichiers ATESS retracent l'itinéraire d'une mission d'un train et de ce fait, constituent les documents de base pour l'instruction des enquêtes judiciaires à la suite d'un accident. En raison de son rapport direct avec la sécurité, le travail du vérificateur revêt une importance particulière pour le contrôle à posteriori des dysfonctionnements de conduite.

Voici un exemple de fichier ATESS que le vérificateur est amené à analyser. Un fichier ATESS contient un enregistrement des espaces parcourus, un enregistrement des temps, un enregistrement de vitesses puis enfin un enregistrement d'autres événements de conduite. On lui associe un fichier lpb qui est une visualisation graphique des fichiers ATESS.



01923 dist:005851 05/03/03 13:10:30 FR7102837A SP 860314 CHI TO

Tout | Rien | Base | EL | Série

Rang	Dist/Vit.	Date/Heure	
02271	009800 005		[0B] Seuil vitesse
02272	009800 002		[0B] Seuil vitesse
02273	009800 000	05/03/03 13h 50'34	[08] Arrêt
02274	009800 000	05/03/03 13h 51'44	[99] MP(CO)I sur la position 0 ou MP(TT)F sur la position 0 ou Fre
02275	009800 000	05/03/03 13h 51'44	[AD] Effort traction Non Nul par MP(CO)I ou par VI ou par MP(TT)F
02276	009800 000	05/03/03 13h 51'50	[07] Mise en mouvement
02277	009800 003		[0B] Seuil vitesse
02278	009800 005		[0B] Seuil vitesse
02279	009800 008		[0B] Seuil vitesse
02280	009800 010		[0B] Seuil vitesse
02281	009801 013		[0B] Seuil vitesse
02282	009801 015		[0B] Seuil vitesse
02283	009802 018		[0B] Seuil vitesse
02284	009802 020		[0B] Seuil vitesse

Figure 1 : Fichier ATESS

[3] Agent chargé de vérifier les enregistrements des signaux/vitesses en se basant sur les DSV.
[4] Représentation graphique des fichiers ATESS.

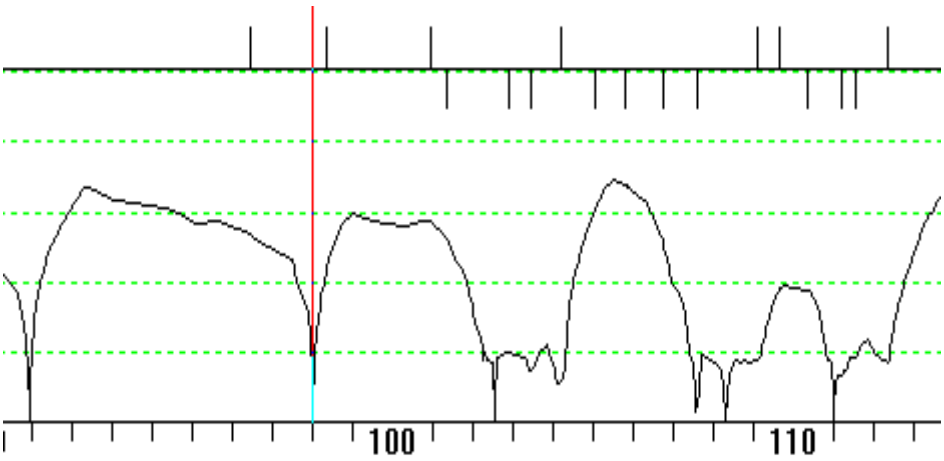


Figure 2 : Représentation graphique de fichier ATESS.

La *Figure 1* représente le fichier enregistré par les cassettes ATESS. On retrouve le rang des différents événements de conduite, la distance parcourue en décamètre depuis l'introduction de la cassette, les enregistrements de temps ainsi que le libellé des événements de conduite avec leur identificateur servant à les caractérisés. L'application VisuBG, développé par la Direction de la Traction, permet une représentation graphique (*Figure 2*) de la première figure. Ainsi la *Figure 2* visualise le profile de vitesse avec le kilométrage et les signaux rencontrés.

Bandes Types et Renseignements Techniques

Les bandes types sont des schémas de référence représentant la liste des signaux, des aiguilles, des établissements ou autres points remarquables rencontrés par un train lors de sa progression.

Les bandes types sont élaborées grâce à l'application EPURE. C'est une application fonctionnant sous AutoCAD qui réalise automatiquement les bandes types à partir de fichiers répertoriant les plans de voies et leur infrastructure.

Ces données sont stockées dans des fichiers d'extension « bdt [5] » qui sont utilisés par la base de données décrivant un DSV.

Les RT sont des documents fournissant entre autres les vitesses limites en fonction des caractéristiques des tronçons de voie. Par ailleurs, les RT contiennent aussi des informations sur l'infrastructure des voies.

Exemple de Bande Type :

[5] Format des fichiers décrivant une bande type.

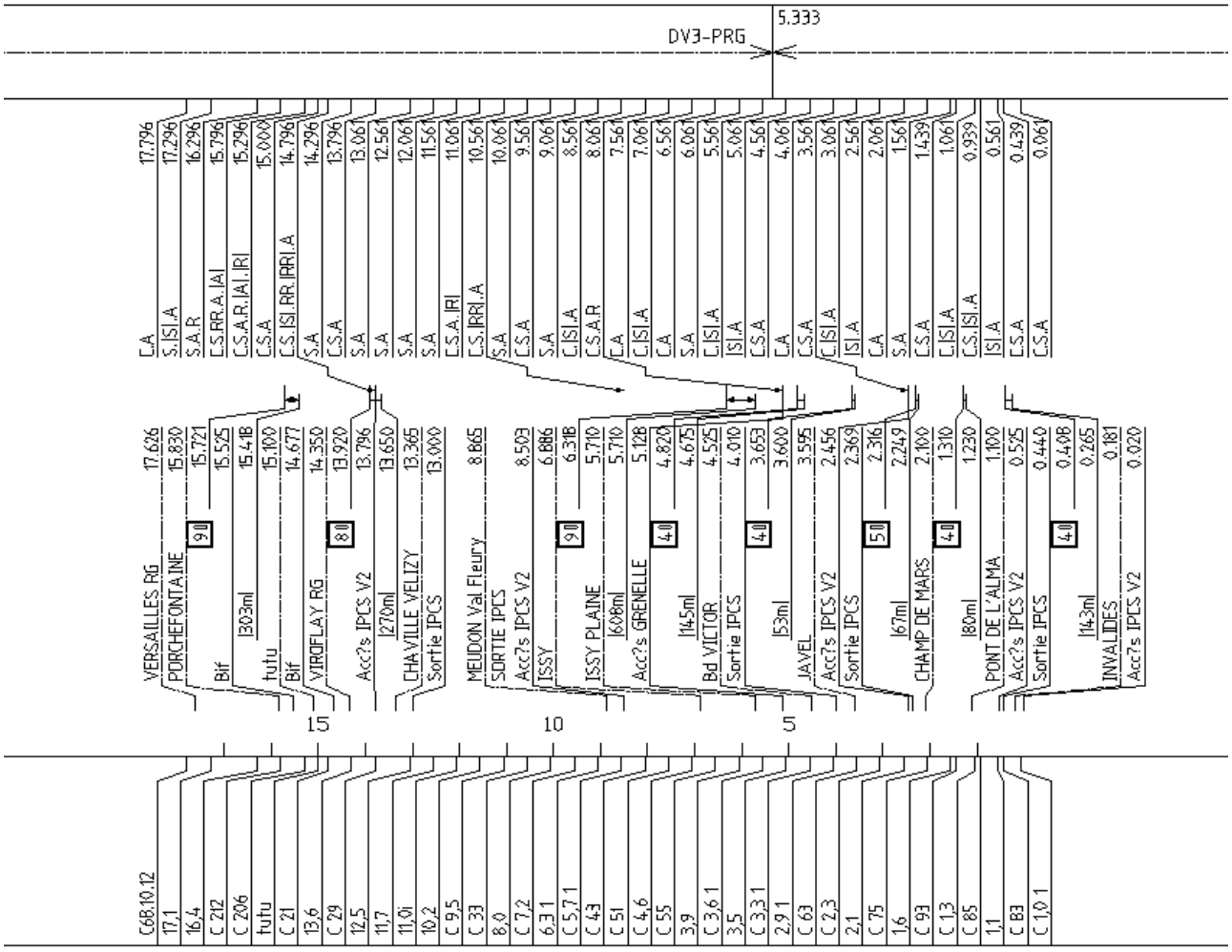


Figure 3 : Bande Type

La partie inférieure de la bande type représente l'implantation et la désignation des signaux. La partie supérieure représente les indications portées par les signaux. La partie du milieu représente les différents établissements, les aiguillages ainsi que leur signal annonceur et aussi les zones de limitation de vitesse.
Une Bande Type peut être issue de plusieurs fichier bdt.

Notion de Mission

A chaque mission correspond un fichier ATESS. Les missions sont des trajets effectivement réalisés par les trains. Ils sont définis par une gare origine, une gare destination et un itinéraire. Une mission peut être décrite par des portions de plusieurs DSV (Diagrammes Signaux/Vitesses). Ce sont les zones de vitesse limite des DSV associés à ces missions qui seront utilisées par la suite lors de la vérification des données ATESS.

Diagrammes Signaux/Vitesses

Un DSV est composé généralement d'un cartouche (partie gauche du dessin) précisant des caractéristiques nominales du parcours, d'une partie représentant les voies avec les signaux, d'une partie donnant les zones de limitation de vitesses selon le type de train, puis enfin d'un axe de kilométrages avec des points remarquables (établissements).

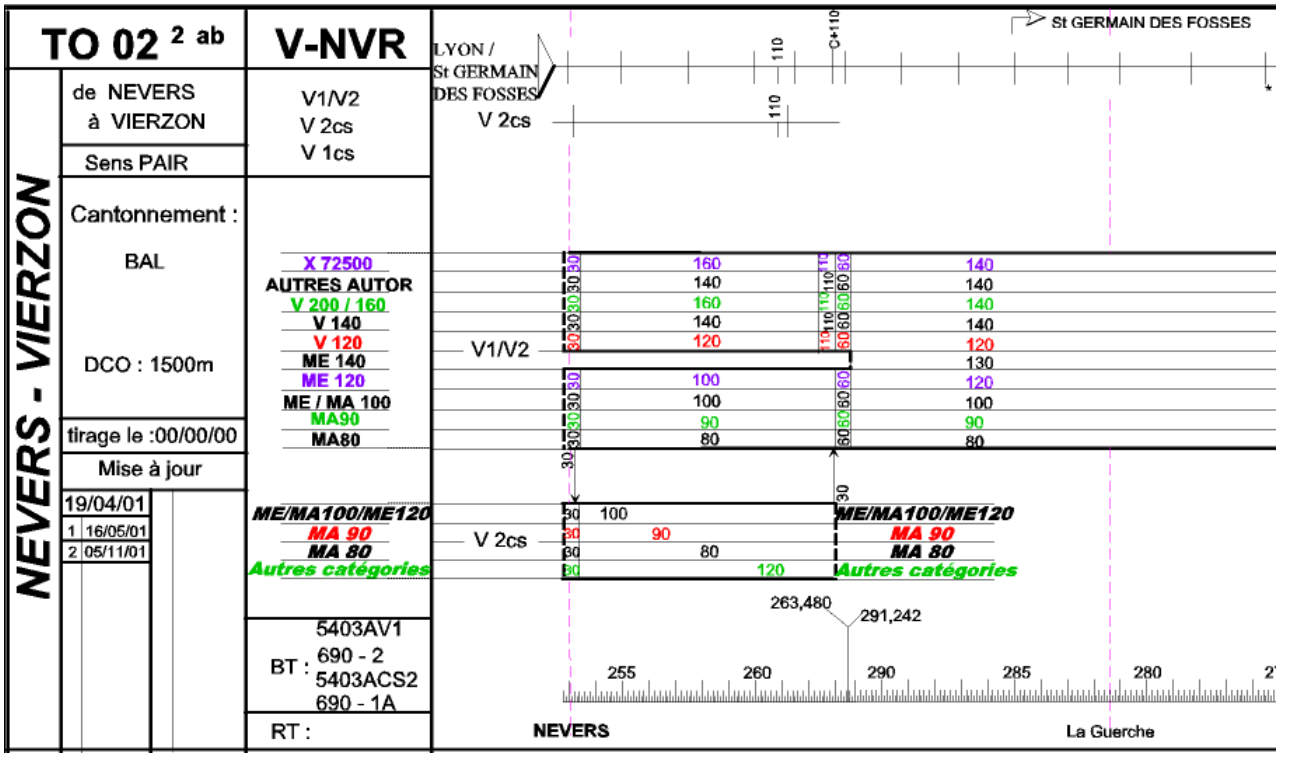


Figure 4 Diagrammes Signaux/Vitesses

Le but de mon stage est de réaliser le diagramme précédent sachant qu'il est entièrement décrit dans une base de données ACCESS. Je reviendrais par la suite sur chaque composant de ce diagramme en détail.

Quelle est l'utilité de ce Diagramme ?

La mission effectuée par un train est réellement décrite par le profile de vitesses issues des fichiers ATESS (*Figure 2*). La superposition de ce profile avec des parties bien choisies de quelques DSV permet de retracer en détail l'itinéraire du train et d'analyser les comportements de conduite sur les différents tronçons de voies.

Cahier des charges fonctionnel

Mon travail consistera à créer une DLL[6] s'intégrant à AutoCAD. Cette DLL joue le rôle d'interface entre le mapping objets relationnels de la base de données décrivant les DSV et les éléments graphiques d'AutoCAD. L'API ObjectARX d'AutoCAD permet de créer une DLL capable d'être exécuté dans un environnement AutoCAD et donc capable de dessiner les DSV.

Vue générale

L'utilisateur de l'application devra choisir à travers une IHM accessible à partir d'AutoCAD, un DSV parmi la liste des DSV existant en base de donnée. La validation de son choix avec des options éventuelles, induira automatiquement le dessin sur AutoCAD du DSV. C'est à dire le dessin du cartouche, les voies et les signaux, les tableaux de zone de vitesse, les connections entre les voies et enfin les points kilométriques avec les points remarquables du parcours (aiguillages et établissements).

6 Dynamique Link Library : Bibliothèque de liaison dynamique Windows

Le centre de vérification de Tours-St Pierre et la Direction de la Traction, nous ont fourni un document de référence répertoriant les caractéristiques graphiques du DSV : « Confection d'un DSV informatisé ». Ce document n'a pas était validé par les centres de vérification mais constitue un support important pour mon application.

Les besoins fonctionnels sont repartis en unités logiques de besoins. Chacune correspond à une unité de dessin du DSV. Les unités de dessin seront présentées selon la spécification décrite dans le document fourni par le centre de vérification et selon les DSV déjà existant dessiné sous Designer(voir *Figure4*). Les deux spécifications diffèrent dans certains points.

Positionnement du cartouche

Le cartouche concerne la partie gauche du DSV (*Figure 5*). Il contient des informations concernant le parcours, le centre émetteur du document, la catégorie de train ainsi que des références vers les documents techniques de base qui ont permis d'établir ce DSV.

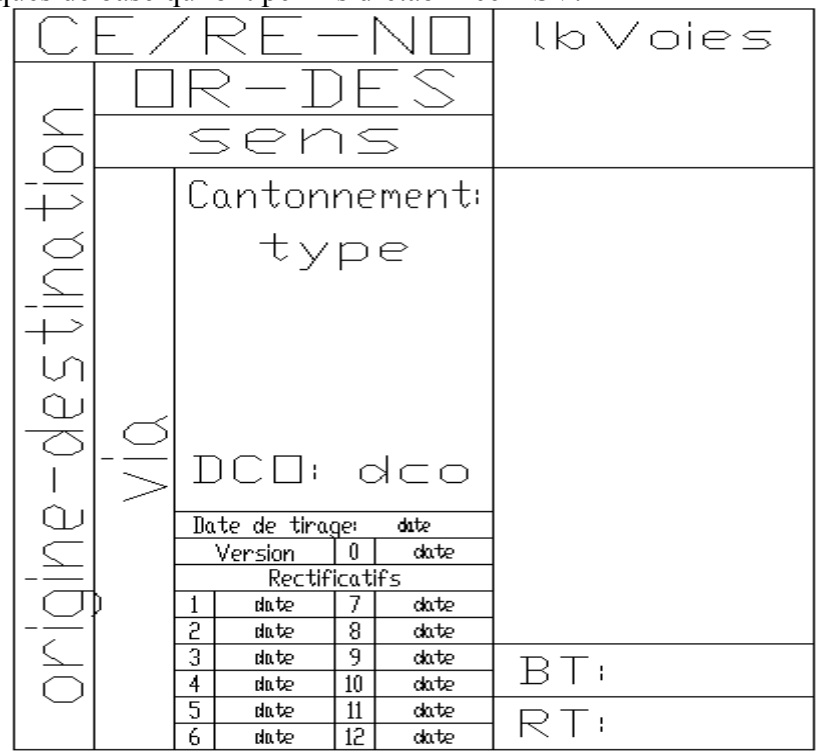


Figure 5 Cartouche du DSV

La figure ci-dessus n'est pas à l'échelle. Dans le document fournit par le centre de vérification, les tailles et les différentes polices de caractères sont bien détaillées.
Toutes les informations présentes dans ce cartouche existent en base de données. Dans la partie Réalisation -Description de l'existant, on verra en détail le modèle de données associé.

Description :

- **CE/RE-NO :** Cette partie regroupe les initiales du Centre Emetteur, de la Région du parcours et du Numéro d'Ordre. Le Numéro d'Ordre permet d'identifier un DSV pour le Centre Emetteur.
- **Origine-destination :** Etablissement d'origine et établissement de destination du DSV. Notons que ces libellés représentent une mission d'un train, cependant il existe bien sûr des missions décrites par un ensemble de parties de plusieurs DSV. On retrouve les initiales de ces libellés dans **OR-DES**.
- **Sens :** Il existe un sens paire et un sens impaire pour les voies. En règle générale les voies partant de Paris sont impaires et ceux arrivant à Paris sont pairs.

- **Via :** Etablissement principal de passage.
- **Cantonnement :** Types de sémaphore utilisé dans les différentes voies pour le partage de la ressource voie par les trains.
- **DCO :** La Distance de Couverture d'Obstacle oblige le train, lors d'un arrêt imprévu, à respecter une vitesse assez faible avant une reprise normale du trafic.
- **LbVoies :** Labels des voies du DSV.
- **Date de tirage/Version/Rectificatifs :** Permet de dater les différentes mises à jour.
- **BT/RT :** Références vers les Bandes Types et les Renseignements Techniques qui ont permis l'élaboration du DSV.

Positionnement des voies et des signaux

Les voies et les signaux doivent être affichés sur la partie supérieure du DSV. Il y a différents types de signaux avec des étiquettes. Les voies sont des lignes horizontales. Les paramètres qui permettront de dessiner ces objets sont dans la base ACCESS.

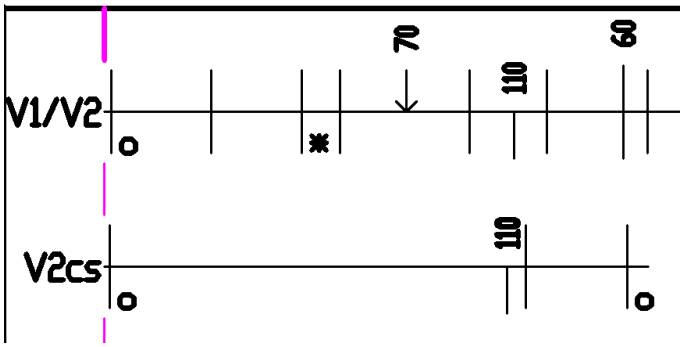
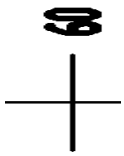


Figure 10 Signaux et Voies

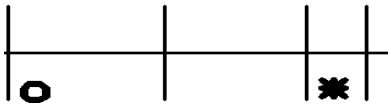
Description :
Une voie est une ligne horizontale dont les dimensions sont à remonter de la base de données.
Il y a quatre types de signaux à représenter dans le cahier des charges initiales:

- **Tiv Fixe *:*** Tableau Indicateur de Vitesse. Il est dit fixe car la vitesse est constante :

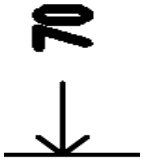
- **Tiv Mobile :** Tableau Indicateur de Vitesse variable :



- **Signal Répété** : Signal visualisé par les mécaniciens (conducteurs) et répété en cabine à l'aide d'un équipement spécifique sur les voies appelé crocodile, on parle aussi de signal crocodile. Il y a plusieurs types de signaux répétés. Ceux à représenter dans le DSV sont : le feu rouge ou rouge clignotant représenté par : *, le feu blanc ou blanc clignotant représenté par : °. Le feu rouge simple est appelé sémaphore, et commande au mécanicien de s'arrêter avant le signal. Le feu blanc autorise le mécanicien en marche de manœuvre, il est donc autorisé en marche à vue. Ce signal ne se rencontre que sur les voies de manœuvre (garages, dépôt). Les autres types de signaux répétés sont aussi représenté mais sans aucune indication.



- **Signal Non Répété :** Signal visualisé par les mécaniciens (conducteurs) et non répété en cabine. Il présente une vitesse conseiller :



Au début de chaque voie, on spécifie son label.

Positionnement des tableaux de zone de vitesse et des connections entre voies

Les tableaux de zone de vitesse sont des rectangles indiquant la vitesse limite à ne pas dépasser en fonction du tronçon de voie et d'un critère de tonnage des trains qu'on appelle indice de composition qui spécifie la catégorie des trains. Ces rectangles se trouvent au centre du DSV dans un emplacement spécifié par le document « Confection d'un DSV informatisé » . Les zones de vitesse sont regroupées par voie d'appartenance et les connections entre voies se trouvent représentées par des flèches entre les blocs de zones.

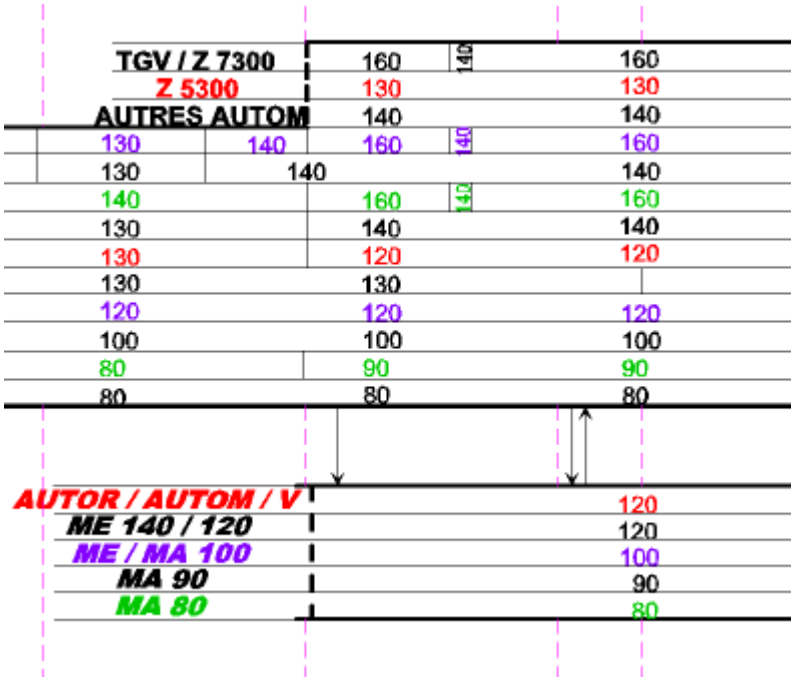


Figure 11 Zones de vitesses et connections (DSV déjà existant)

Il est prévu de mettre les indices de composition en début de zone. Par contre, contrairement à la figure précédente, il y aura une alternance entre deux couleurs au lieu de plusieurs, le noir et le vert.On est amené aussi à trouver une solution en cas de petite zone de vitesse pour afficher la valeur de la vitesse. Il faudrait aussi résoudre le problème en cas de grande densité des connections. D'autres problématiques interviendront lors de la réalisation.

Positionnement des Etablissements et des Sauts de Points Kilométrique (pk)

Les sauts de pk et les établissements sont représentés dans la partie inférieure du DSV.

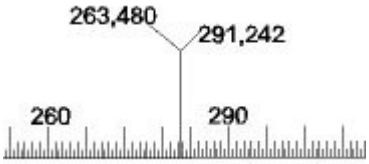


Figure 12 Saut de pk

Un saut de pk est un point de la mission qui possède plusieurs kilométrages puisqu'il est repéré avec des repères différents.

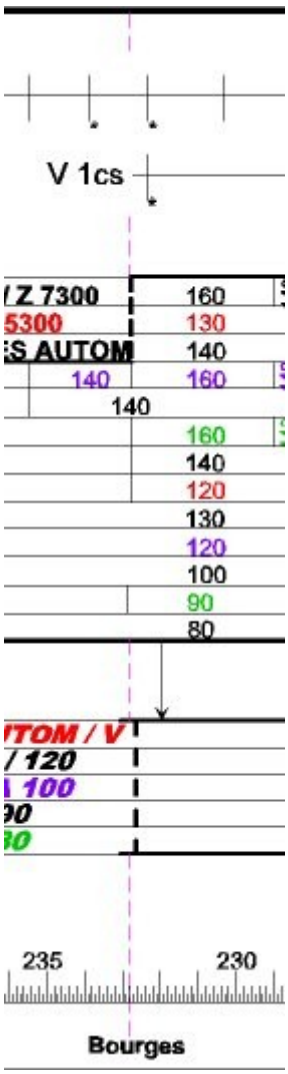


Figure 13 Etablissement

L'établissement Bourges est représenté par son libellé en bas du DSV. Un trait en pointillé permet le repérage de l'établissement sur la hauteur du DSV. Ce trait ne doit pas apparaître sur les objets du DSV.

Positionnement de la graduation des pk

Cette graduation se trouve dans la partie inférieure du DSV. Cette partie du dessin est assez délicate puisque, comme on le verra dans la partie réalisation, il n'existe pas d'objet métier dans le modèle de données représentant la graduation, elle n'est induite que par le pk des établissements et les sauts de pk.

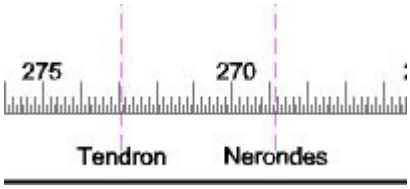


Figure 14 Graduation

Critères de validation du projet

Le centre de vérification de Tours nous a fourni le calque du DSV Nevers-Vierzon via Bourges. Pour valider mon application et étant donné que l'IHM remplissant la base de données des DSV n'a pas encore été développée, on a créé un DSV représentant le trajet Nevers-Bourges à l'aide d'un fichier SQL. Ainsi, il existe dans la base de données un DSV que l'on pourra dessiner et comparer par la suite avec le calque.

A part ce test de conformité de haut niveau, il nous a été difficile de définir une stratégie de test automatisée permettant de scruter les éléments graphiques et valider leur position.

Moyens mis à disposition du stagiaire

Les moyens mis à ma disposition sont : un bureau avec un poste de travail contenant tous les outils nécessaires au développement. J'utilise Visual C++ sous environnement Windows NT pour le développement de l'application à intégrer sous AutoCAD. J'utilise par ailleurs les outils suivant : AutoCAD 2002, ObjectARX(API objet d'AutoCAD), Access 97, WinCVS (gestionnaire de version), CppUnit, RationalRose (modélisations objets).

Planification du projet

Les trois premières semaines du stage ont été consacrées à la familiarisation aux outils de travail. Notamment avec le mapping objet relationnel de la base de données des DSV et l'API ObjectARX d'AutoCAD.

Après chaque conception et réalisation d'une unité logique de besoin fonctionnel, j'effectue une livraison de code sous CVS. La procédure suivante est systématiquement suivie :

- Mise à jour de la version locale.
- Compilation des différents projets.
- Validation des tests existants déjà.
- Livraison du code sous CVS.

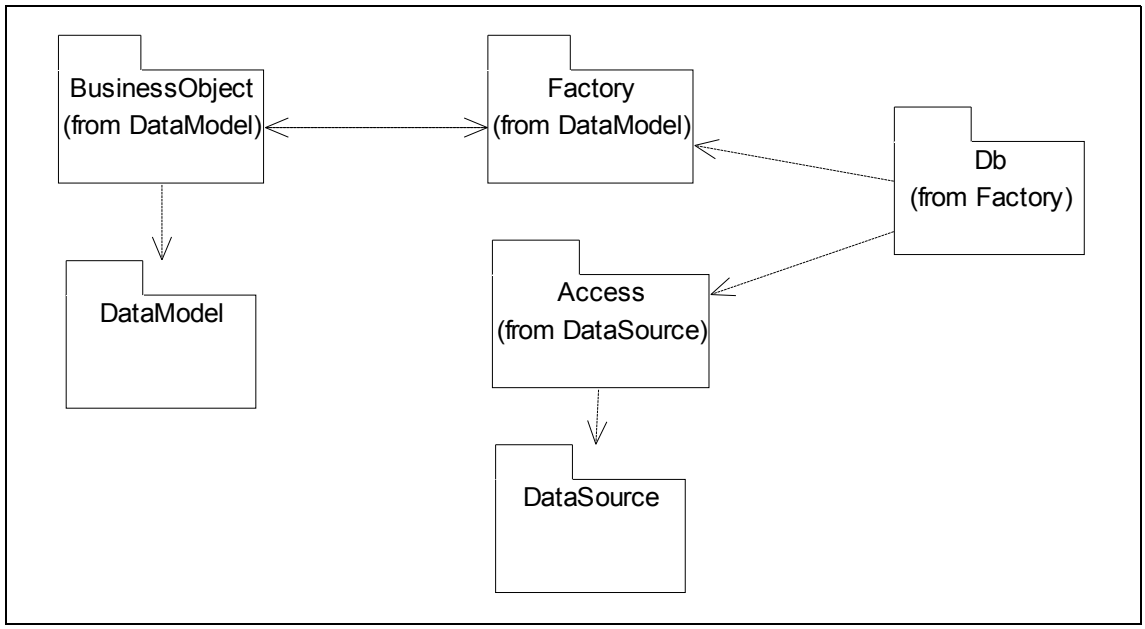
PARTIE N°4.RÉALISATION DU PROJET

J'ai utilisé pour la réalisation de mon application plusieurs modules déjà existant. Leur description détaillée serait fastidieuse et sans intérêt. Cependant les relations existant entre ces modules doivent être explicitées.

Un module est aussi appelé un composant logiciel. Dans notre projet, c'est un espace de nom au sens C++, c'est à dire un namespace.

Description de l'existant

Vue globale



La notation A ---------->B veut dire A utilise B.

Cette architecture permet de mapper la base de données Access des DSV.Dans **BusinessObject**, on trouve tous les objets métiers. A chaque table de la base de données correspond un objet métier. A chaque objet métier est associé une classe **Factory** qui est chargée de réaliser la sauvegarde ou le chargement en base de données. Ces factories utilisent des requêtes SQL pour

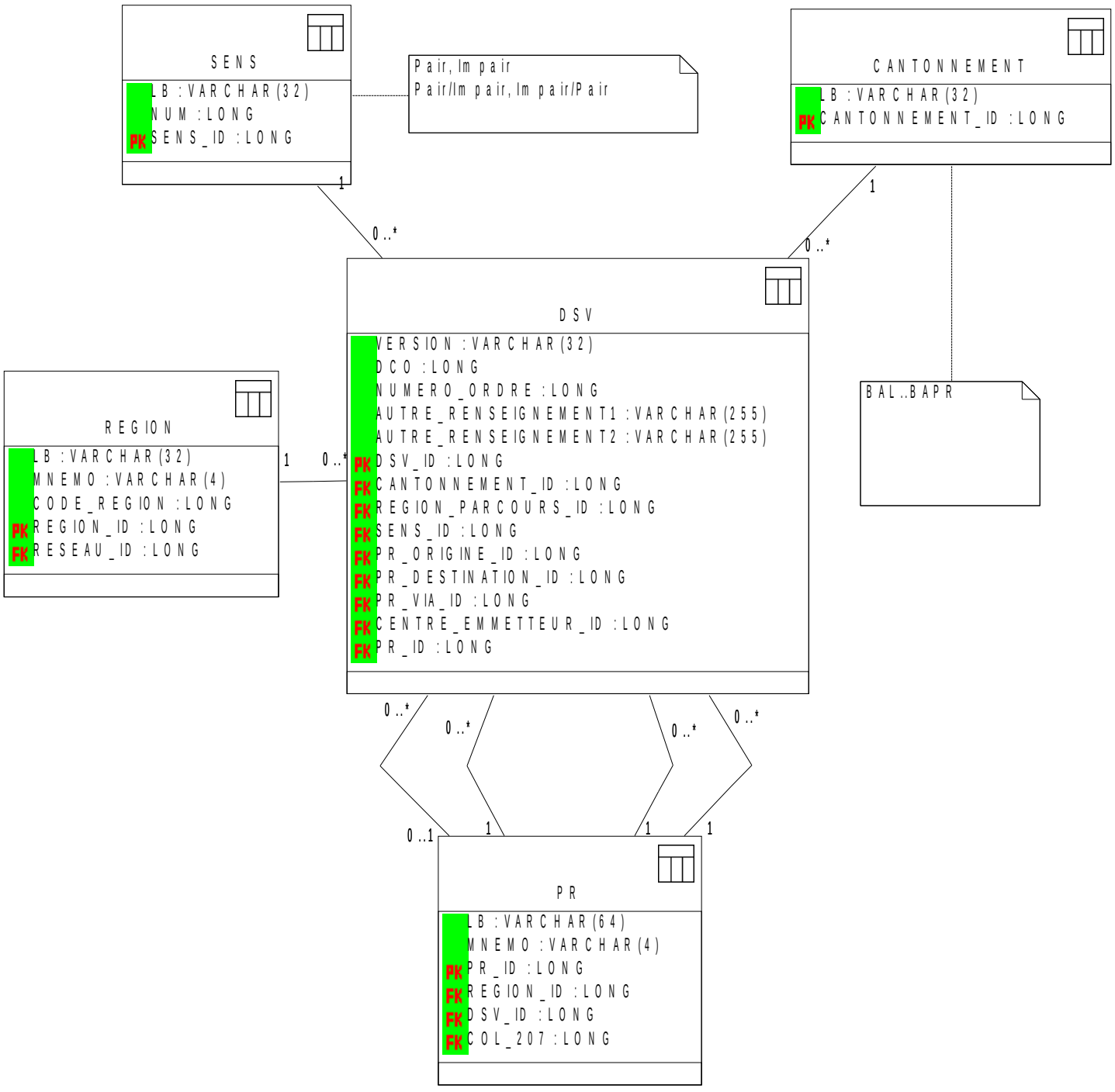
accéder à la base de données. Cet accès est délégué à la classe **AccessDatasource** du module Access qui utilise la technologie DAO (Data Access Object) pour s'interfacer avec Access97.
Il est nécessaire, avant d'aborder les composants développés par moi-même d'explicité le modèle de données des objets métiers.

Modèle de données

Ce modèle se compose de trois parties. Une permettant l'identification d'un DSV, une décrivant les mises à jour et les différentes sources permettant le dessin du DSV et une dernière modélisant les objets du DSV c'est à dire signaux, voies, zones de vitesse, établissement, saut de pk.

- **Identification**

~~Cette partie concerne essentiellement le cartouche du DSV.~~



Ce modèle relationnel permet d'avoir plusieurs informations présentes dans le DSV. Les relations sans intérêt pour mon application n'ont pas été représentées. Par ailleurs, il y a certains attributs qui me seront inutiles.

- **La table DSV**

Cette table regroupe quasiment toutes les informations nécessaires pour dessiner le cartouche du DSV : la distance de couverture d'obstacle (DCO), le numéro d'ordre ainsi que d'autres clés étrangères permettant de récupérer les autres informations. On retrouvera cette table dans les autres parties du modèle.

- **La table PR**

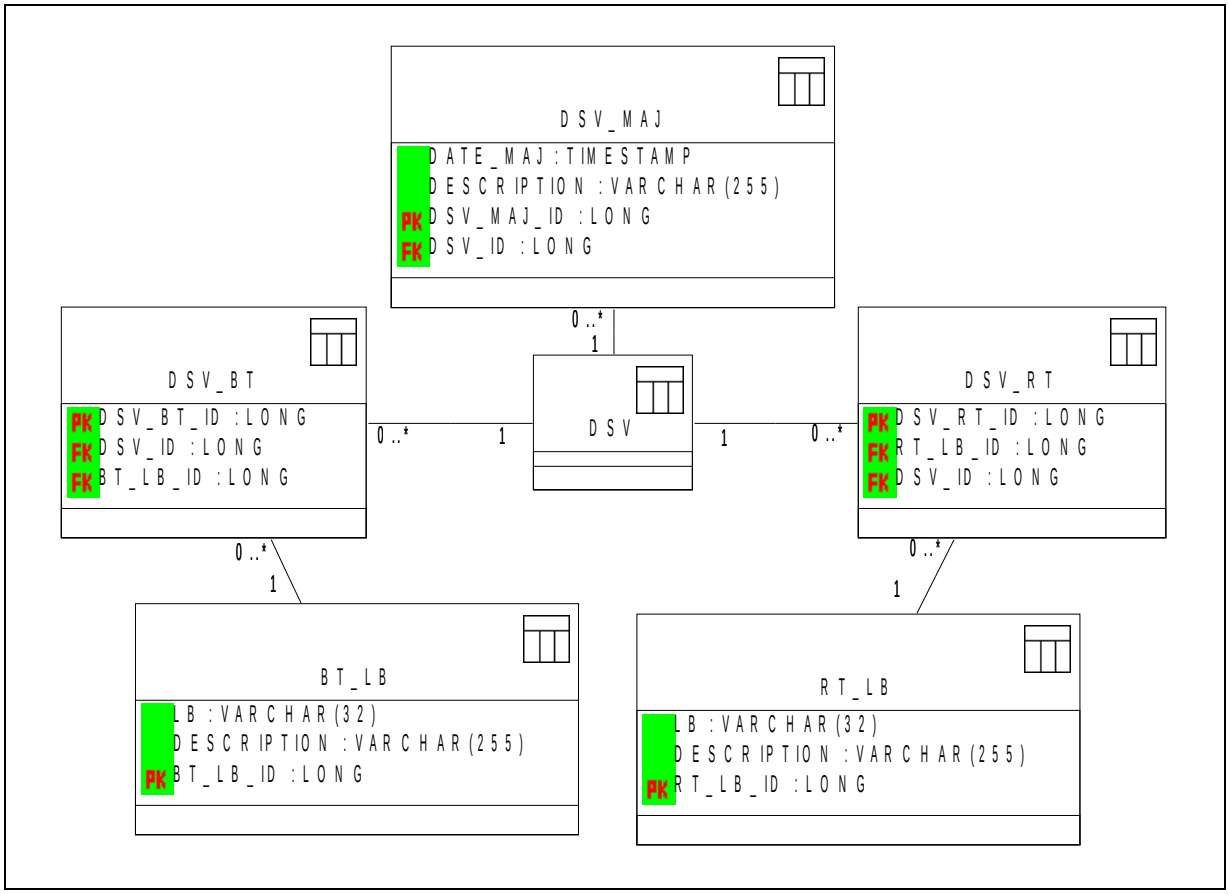
Elle regroupe tous les points remarquables de l'ensemble du réseau. C'est à dire les établissements susceptibles d'être l'origine, la destination, le via ou le centre émetteur d'un DSV. Les cardinalités montrent par exemple qu'ont peut omettre de préciser le via d'un DSV.

- **Les tables SENS, CANTONNEMENT, REGION**

Ces tables enregistrent les libellés présents dans le cartouche : le sens du parcours, le type de sémaphore utilisé par le cantonnement et la région du parcours.

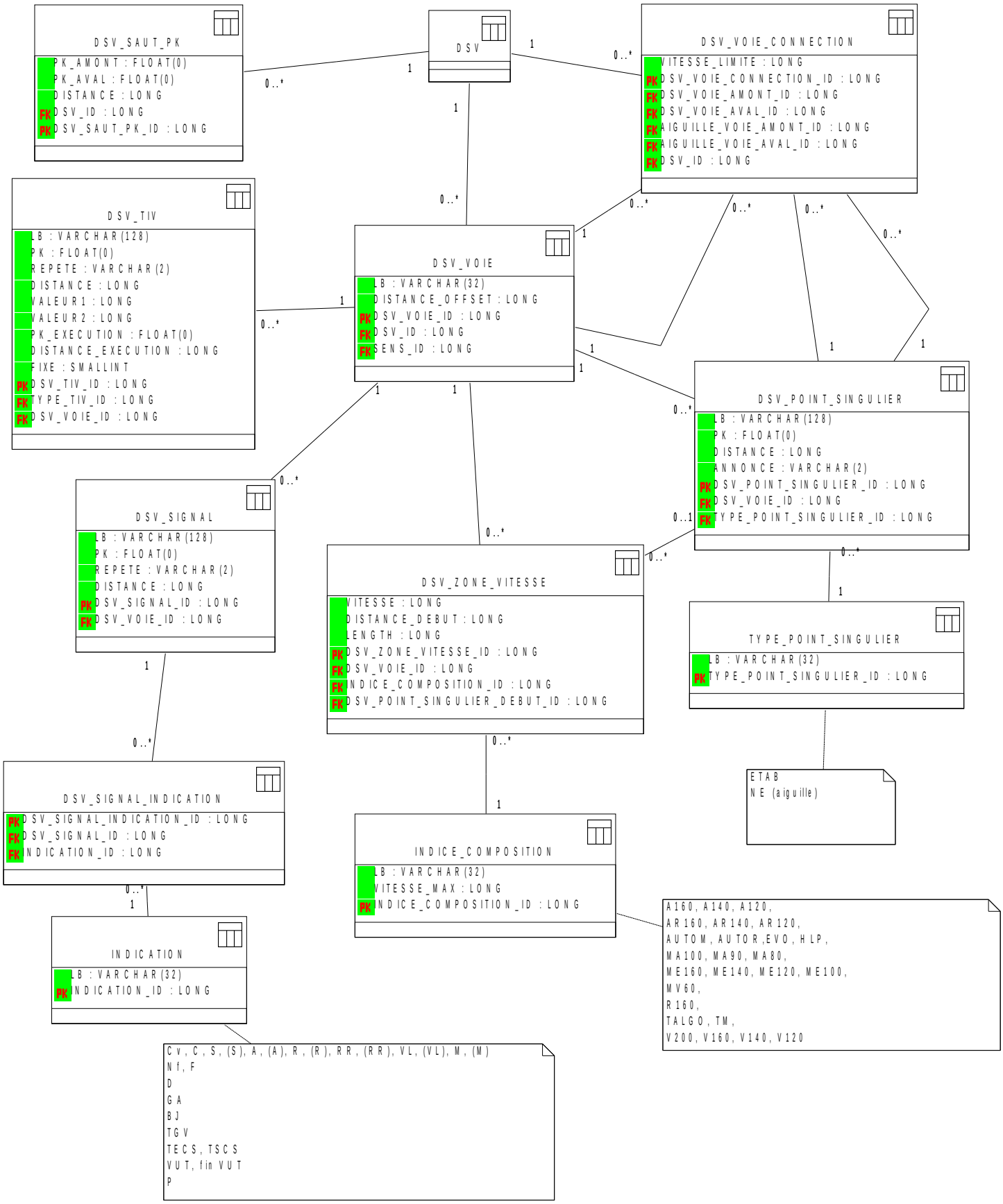
- **Mise à jour**

Cette partie concerne aussi le cartouche du DSV.



Ce modèle permet de préciser les dates de mise à jour du DSV dans le cartouche ainsi que les libellés des bandes types et des renseignements techniques utilisés pour réaliser le DSV.

- **Description des objets du DSV**



On ne décrira ci-dessous que les informations utiles pour le dessin du DSV.

- **La table DSV_VOIE**

Cette table contient les libellés des voies du DSV (LB) ainsi que leur distance par rapport à l'établissement d'origine (DISTANCE_OFFSET).Une clé étrangère spécifie le DSV contenant la voie (DSV_ID). Une voie ne connaît pas pour l'instant sa longueur. Cette information est présente dans le mapping objet de la table.

- **La table DSV_TIV**

Cette table enregistre les tableaux indicateur de vitesse. C'est une signalisation se trouvant sur les voies. Elle peut avoir une ou deux valeurs de vitesse. Ce qui nous est utile dans cette table, c'est le caractère fixe ou mobile du TIV (FIXE), les vitesses (VALEUR1 et VALEUR2), la distance du TIV qui est par rapport à la voie qui le contient (DISTANCE) et bien sûr l'identifiant de la voie (DSV_VOIE_ID).

- **Les tables DSV_SIGNAL, DSV_SIGNAL_INDICATION, INDICATION**

Ces tables permettent de déterminer les caractéristiques des signaux : la distance par rapport à la voie qui les contient (DISTANCE), la répétition du signal en cabine (REPETE) et l'indication portée par le signal.

- **Les tables DSV_ZONE_VITESSE, INDICE_COMPOSITION**

Ces tables donnent les zones de vitesses associées à une voie et à un indice de composition. Rappelons que l'indice de composition spécifie la catégorie de train sur des critères de tonnage. On a connaissance par ailleurs de la vitesse à respecter dans la zone (VITESSE), la distance de début de zone par rapport à la voie qui la contient (DISTANCE_DEBUT) et la longueur de la zone (LENGTH).

- **La table DSV_VOIE_CONNECTION**

Les connections sont représentés dans le DSV entre les zones de vitesses associées à une voie. La vitesse limite de franchissement de la connexion est donnée par le champ VITESSE_LIMITE. Le positionnement des connections est possible grâce aux différentes clés étrangères pointant vers la voie amont, la voie aval, l'aiguille amont, l'aiguille aval. Les aiguilles sont dans la table **DSV_POINT_SINGULIER**.

- **Les tables DSV_POINT_SINGULIER, TYPE_POINT_SINGULIER**

Cette table enregistre les établissements et les aiguilles de chaque voie. On a ainsi toutes les informations nécessaires pour représenter les établissements des DSV.

- **La table DSV_SAUT_PK**

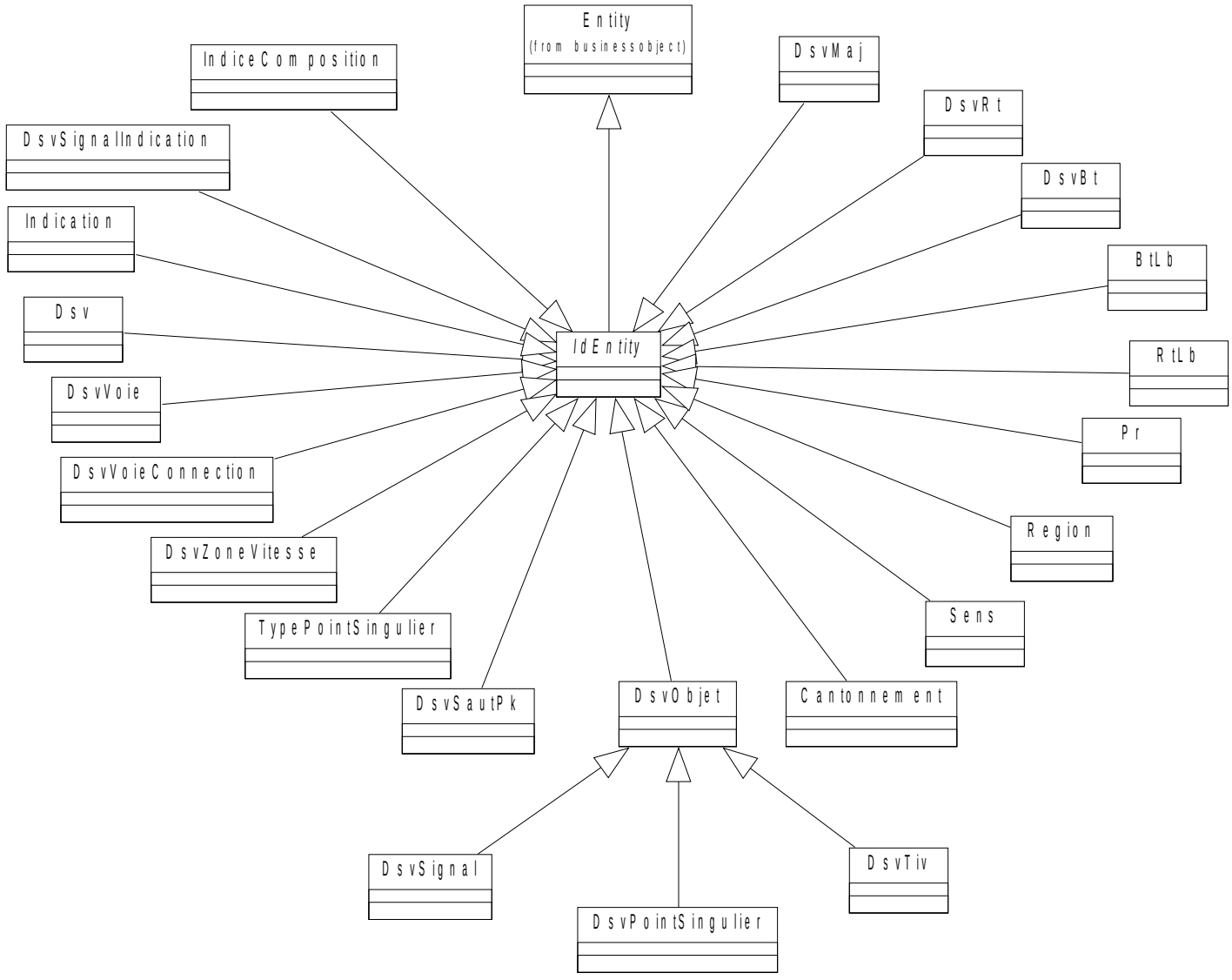
Elle enregistre les sauts de PK du trajet. On a donc sa distance d'implantation (DISTANCE) par rapport cette fois à l'établissement d'origine, les pk amont et aval ainsi que le DSV concerné.

En résumé, les informations permettant de réaliser le dessin du DSV sont tantôt attachées à la table DSV, tantôt au voies du DSV. Ce modèle enregistre ces informations mais n'a pas l'intelligence nécessaire pour les récupérer. C'est le diagramme de classe associé à ce modèle, c'est à dire le mapping objet relationnel qui possède cette intelligence. En terme de composant logiciel, c'est le module BusinessObject qui est charger de faire ça.

Description du composant logiciel BusinessObject

- **Architecture du mapping des objets métier**

Tous les objets métiers sont vus comme une entité : Entity. C'est la classe de base de l'architecture. Une entité est associée à une table du modèle de données. De cette classe, hérite la classe IdEntity qui gère les clés primaires des tables du modèle.

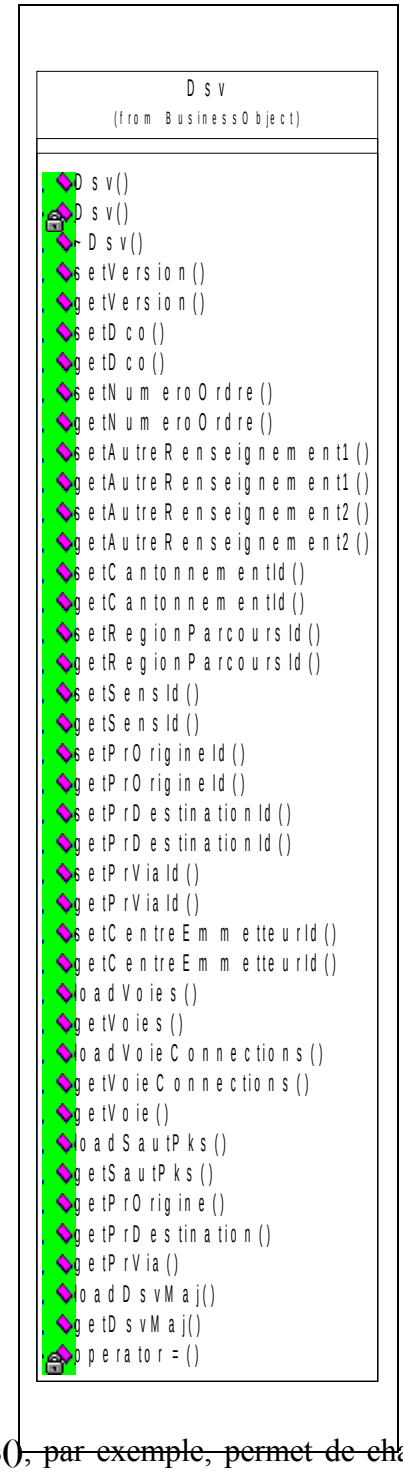


Pour comprendre ce diagramme je vais décrire quelques classes.

- **La classe Dsv**

Les membres de cette classe donnent accès aux informations existant dans la base de données. Cette classe est dotée d'une intelligence qui lui permet de charger des éléments qui existent en base. Ceci

s'explique grâce à la relation existant entre le composant logiciel **Factory** et le composant logiciel **BusinessObject**.

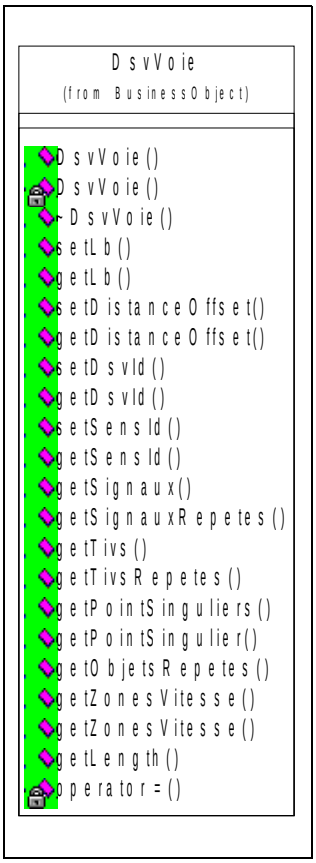


La fonction membre **loadVoies()**, par exemple, permet de charger dans un conteneur membre de Dsv les voies du DSV. Le mapping fait de même pour les sauts de pk : **loadSautPks()**, les connections : **loadDsvConnections()**, et les mises à jour : **loadDsvMaj()**.
Les informations propres à la classe sont chargées par le programme principal de l'application.

- **La classe DsvVoie**

Cette classe mape les informations associées aux voies du DSV. Notamment les Signaux, les Tiv, les zones de vitesse. Cette classe fonctionne de la même façon que la classe Dsv. Cependant on ne trouvera pas de fonction membre du style **loadTiv()** par exemple, pour la simple raison que cette classe cache, en utilisant une classe interne, les méthodes de chargement et permet de faire des

chargements une fois pour toutes à l'aide de booléen. Ce choix a été fait ici par soucis d'optimiser les accès en base.



Ainsi, par exemple, au premier appel **getTivs(),** les tableaux indicateurs de vitesse sont chargés une fois pour toute.

Les autres classes suivent le même modèle.

Ce mapping est la source de données principales qui me permettra de réaliser automatiquement le DSV sur AutoCAD. Les détails permettant de positionner les éléments du DSV seront vues dans les parties qui suivent. On verra aussi comment lier cette source de données, qui est le mapping, avec les objets graphiques d'AutoCAD.

Analyse du cahier des charges et de ses contraintes

Vue générale

Deux principales contraintes se dégagent lors de l 'analyse du cahier des charges. La première concerne la faisabilité de la représentation graphique sous AutoCAD du mapping du modèle objet. Ceci impose une stratégie d'extraction de donnée métier et de placement d'objets graphiques AutoCAD bien précise. La deuxième concerne la lisibilité du DSV après son tracé. En effet la forte densité de certains éléments du DSV risque de rendre ce dernier illisible pour le vérificateur.

Positionnement du cartouche

Cette partie ne pose pas de problème puisque l'information à mettre sur le dessin est essentiellement du texte dont le placement est statique par rapport au cartouche.

Par ailleurs AutoCAD permet de créer un modèle sous forme de bloc avec des attributs textes qu'on peut récupérer et assigner avec la bonne valeur.

Positionnement des voies et des signaux

Le problème susceptible de se poser ici est celui de la forte densité des signaux et des Tiv. Le cahier des charges prévoit de mettre les différentes indications verticalement pour éviter au mieux la confusion entre différents signaux ou Tiv. Vu que dans le document fourni par le centre de Tours, la hauteur du DSV est limitée à 140 mm, il nous est impossible d'afficher plus de trois niveaux de voies. Ainsi, dans le cas d'un DSV avec plus de trois niveaux de voie, il faudrait donner la possibilité à l'utilisateur de dessiner les voies qu'il désire. Cette fonctionnalité n'est pas prévue dans le cahier des charges mais peut être ajoutée dans une extension future de l'application. Il faudrait, par ailleurs, définir une stratégie de placement pour les voies, c'est à dire donner un ordre aux voies. A priori, ceci dépend des longueurs de voie, de leur distance de début et de leurs interconnections qui sont représentées entre les zones de vitesse.

En outre, il faudrait faire attention au positionnement des signaux et Tiv car ces derniers sont repérés par rapport à la voie qui les contient.

Positionnement des tableaux de zone de vitesse et des connections entre voies

Cette partie représente beaucoup de contrainte. En effet, les zones de vitesse peuvent être très petites et donc il serait difficile de positionner la vitesse à respecter dans cette petite zone. En outre la représentation d'une zone de vitesse n'est pas homogène sur tout le DSV, elle dépend de sa position par rapport à la voie. Par ailleurs, le nombre de niveaux de zones de vitesse est non seulement conditionné par le nombre de voies mais aussi par le nombre d'indices de composition existant, c'est à dire les catégories de train. Ainsi un grand nombre d'indices de composition ne peut donner un DSV lisible. Dans une extension future de l'application, il faudrait donner le choix à l'utilisateur de dessiner que les indices de composition qu'il désire. Les connections seront représentées entre les zones de vitesse des différentes voies.

Positionnement des Etablissements et des Sauts de Points Kilométrique (pk)

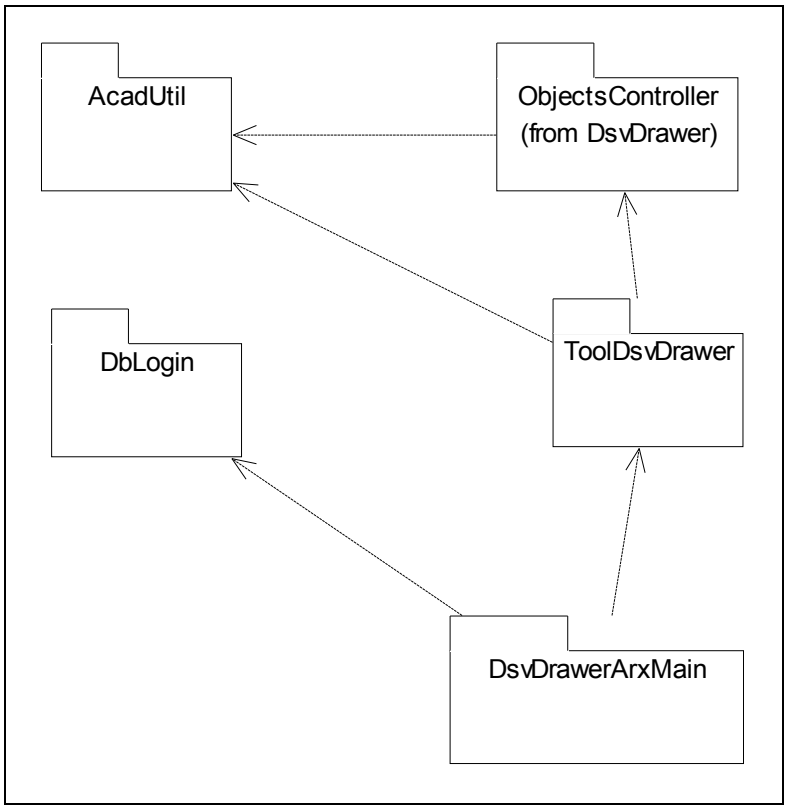
Les sauts de pk ne présentent aucun problème pour leur représentation. Par contre le repérage des établissements qui se fait à l'aide des pointillés en couleur magenta est délicat à réaliser puisqu'il ne correspond pas à un objet métier. C'est le programme principal qui permettra leur réalisation.

Positionnement de la graduation des pk et du cadre général

Cette partie est assez délicate à représenter car la graduation n'existe pas en soi dans le mapping objet, ce sont les sauts de pk et le pk des établissements qui permettront sa réalisation.
Le document fourni par le centre de Tours ne limite pas la longueur du DSV et donc cette dernière sera imposée par la voie se situant la plus à droite du DSV et on pourra ajuster le cadre général du DSV.

Conception générale

Les composants logiciels développés peuvent être séparés en deux groupes, un concernant l'intégration des objets métiers sous forme graphique sous AutoCAD et un permettant la réalisation de l'IHM et de la DLL (en langage AutoCAD on parle de l'ARX) s'intégrant à AutoCAD.



La notation A ---------->B veut dire A utilise B.
Les composants **AcadUtil**, **ObjectsController** et **ToolDsvDrawer** permettent l'interfaçage des objets métier avec les objets AutoCAD (ObjectArx). Les composants **DbLogin** et **DsvDrawerArxMain** définissent la DLL et l'IHM permettant de réaliser le DSV sous AutoCAD.

- **DbLogin**

Le composant **DbLogin** permet de se connecter à la base de données grâce à une boite de dialogue permettant la recherche de fichier après avoir saisi un login et un mot de passe. Ce composant n'a pas été développé par moi-même et utilise d'autres composants qui ne seront pas exposés ici.

- **AcadUtil**

Ce composant sera décrit dans la partie conception détaillée. Il fournit des outils pour insérer des dessins, sous forme de blocs sous AutoCAD.

- **ObjectsController**

Ce composant permet l'association entre un objet métier et un bloc AutoCAD.

- **ToolDsvDrawer**

Ce composant permet l'agencement de tous les objets graphiques du DSV.

- **DsvDrawerArxMain**

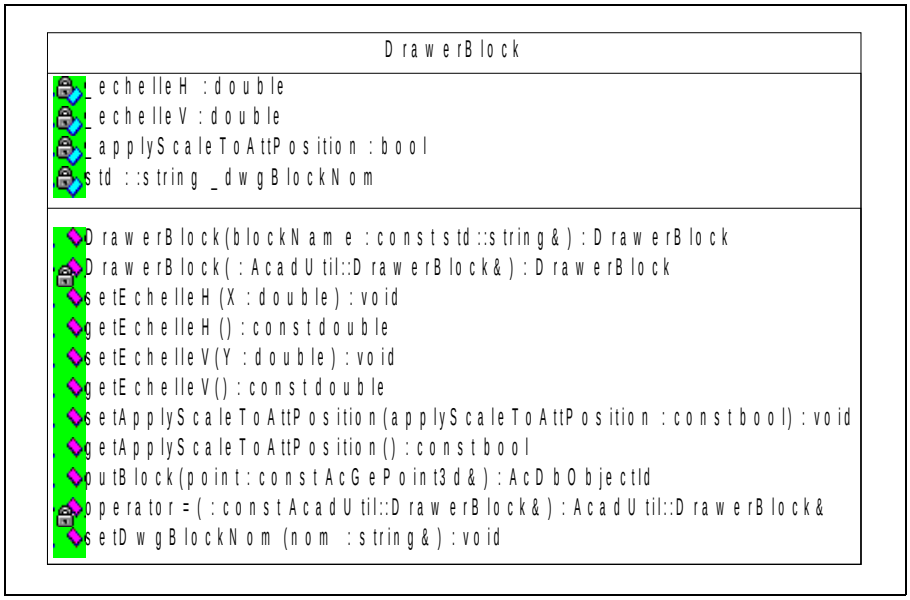
Ce composant permet de charger une liste de DSV pour la présenter à l'utilisateur dans un panneau de dialogue. Il gère les messages entre les éléments de l'IHM. C'est le composant qui définit le point d'entrée de l 'application sous AutoCAD ainsi que la commande AutoCAD qui permet de dessiner le DSV.

Conception détaillée

AcadUtil

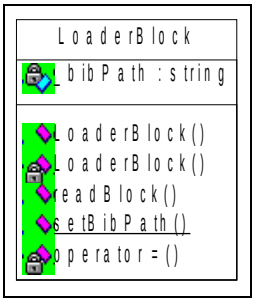
Dans l'espace de nom AcadUtil, il existe trois classes et une fonction utilitaire.

- **La classe DrawerBlock**



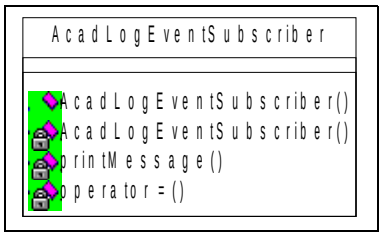
Le constructeur de **DrawerBlock** prend en paramètre le nom d'un bloc existant dans la bibliothèque de blocs. Cette bibliothèque contient des modèles de dessin avec des attributs textuels. Les opérations membres de cette classe fournissent des fonctionnalités sur ce bloc ainsi que sur ces attributs. On peut par exemple changer l'échelle du bloc ou celle de ses attributs.
La fonction membre **putBlock()** permet d'insérer un bloc en spécifiant en paramètre le point d'insertion qui est pour AutoCAD un **AcGePoint3d** . Cette fonction permet d'ouvrir la base de données courantes d'AutoCAD, d'accéder à l'espace de dessin AutoCAD et d'ajouter à cette base une instance du modèle de bloc. Elle renvoie un **AcDbObjectId** qui est un identifiant du bloc inséré. Avec cet identifiant, on pourra accéder à ce bloc est assigner ses attributs en temps voulu.
La fonction permettant l'accès à ces attributs n'est pas une fonction membre car elle peut être utilisée sans avoir à instancier un DrawerBlock.

- **La classe LoaderBlock**



Le constructeur de **LoaderBlock** prend en paramètre le nom d'un bloc existant dans la bibliothèque de blocs (voir en annexe) puis appel la fonction membre **readBlock()** qui, à partir du membre **_bibPath** trouve le chemin de la Bibliothèque de blocs AutoCAD et les charge dans la base de données courantes d'AutoCAD. Ainsi les blocs sont utilisables dans l'espace de dessin AutoCAD.

- **La classe LogEventSubscriber**



Cette classe donne la possibilité d'écrire sur la fenêtre d'AutoCAD à travers la méthode **printMessage()**. Elle servira pour informer l'utilisateur de l'IHM.

- **La fonction getAttribute()**

Cette fonction appartient au namespace AcadUtil. Sa signature est :

AcDbAttribute*
AcadUtil::getAttribute(const AcDbObjectId& id, const std::string& attributeName)

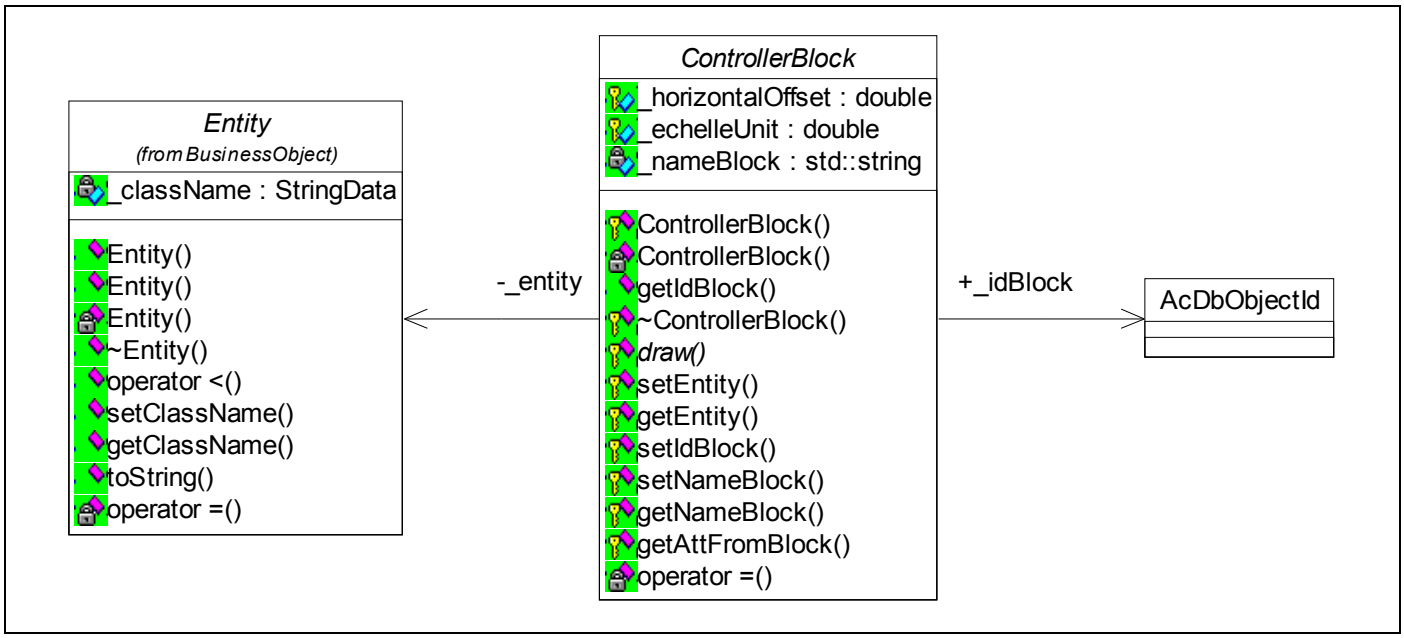
Elle renvoie un pointeur vers un attribut d'un bloc AutoCAD qui existe déjà dans l'espace de dessin. Elle prend en paramètre l'identifiant du bloc concerné et le nom de l'attribut.

Ainsi on a une boîte à outils permettant de dessiner un bloc sur l'espace de dessin AutoCAD et permettant de spécifier la valeur de ses attributs.

ObjectsController

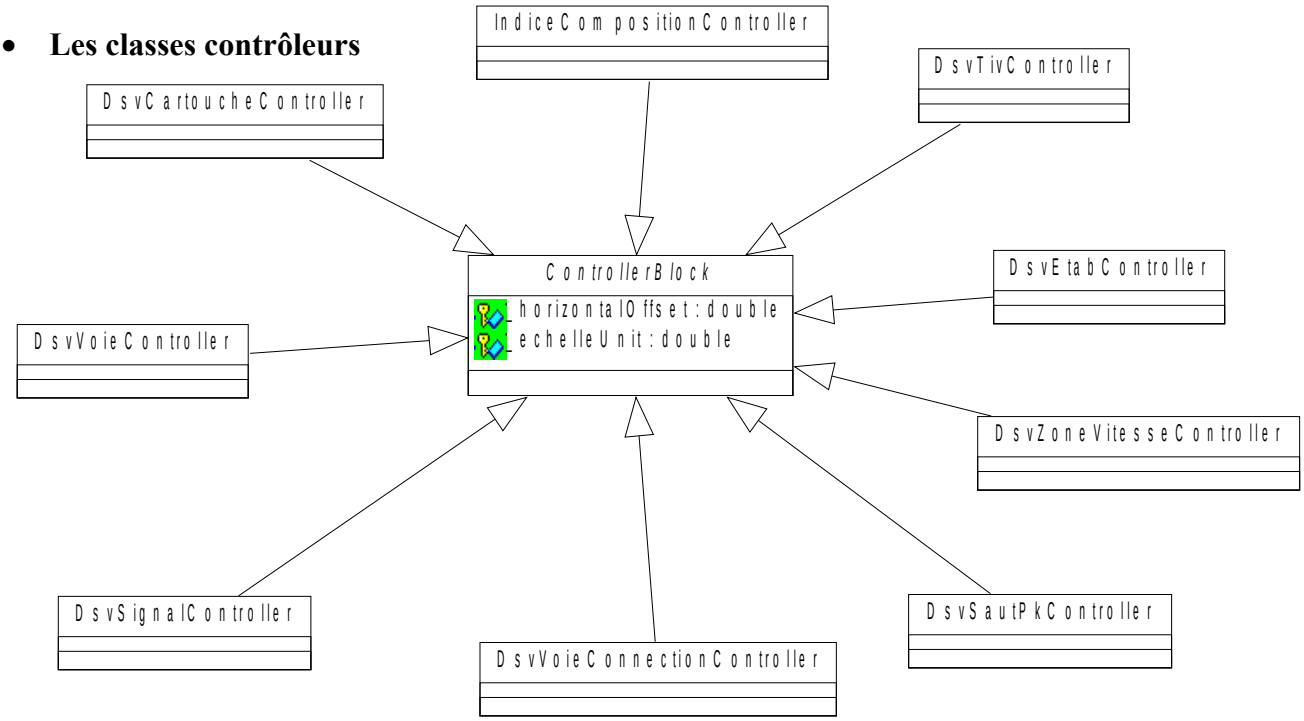
Ce composant lie un objet métier avec un bloc de dessin AutoCAD.
C'est une hiérarchie de classe avec des classes dites contrôleurs dont la classe mère est **ControllerBlock** et de deux classes, dites de spécification, **Offsetable** et **YPositionable** dont quelques contrôleurs héritent, notamment ceux qui gèrent, des blocs dont l'ordonnée et l'abscisse dans le dessin sont variables.

- **La classe ControllerBlock**



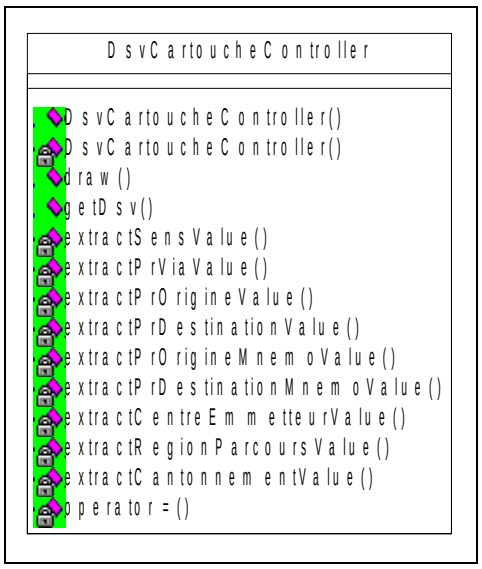
ControllerBlock lie un objet AutoCAD : **AcDbObjectId** avec un objet métier en l'occurrence la classe mère des BusinessObject : **Entity.** Les flèches pleines sont des relations de contenance. **ControllerBlock** possède donc un membre **_entity** et un membre **idBlock.** Cette classe possède des caractéristiques communes à tous les contrôleurs susceptibles d'être définis. Par exemple le nom du bloc qui va être dessiné : **_nameBlock**, l'unité d'échelle du dessin : **_echelleUnit**, le décalage horizontal de tous les objets métiers qui vont être représentés sur le dessin du DSV.
Trois opérations membres de cette classe sont importantes. L'opération **draw()** qui est virtuel pure, c'est à dire qu'on est obligé de définir pour chaque classe héritant de **ControllerBlock.** Cette fonction appelle le **DrawerBlock** du composant **AcadUtil et** dessine le bloc identifié par le membre **idBlock.** L'opération **getAttFromBlock()** qui utilise la fonction utilitaire de **AcadUtil** et permet l'accès aux différents attributs. L'opération **getEntity()** qui permettra aux classes filles de récupérer ~~le bon businessObject~~.

- **Les classes contrôleurs**



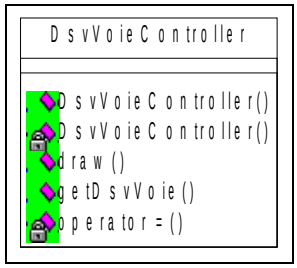
Notons que dans ce diagramme de classe, le contrôleur de la graduation du DSV est inexistant. C'est normal, puisqu'il n'y a pas d'objet métier lui correspondant. La graduation sera traitée ailleurs.
Notons que les contrôleurs ne remontent aucune information de la base de données Access, ce n'est pas leur rôle.

- **DsvCartoucheController**



Le constructeur prend une instance d'**Entity** pour initialiser le membre hérité : **_entity.** Elle définit une méthode **getDsv()** qui utililise la méthode héritée **getEntity()** pour récupérer l'entité de type **Dsv** du mapping via un casting dynamique. Enfin la méthode **draw()** fait appel au DrawerBlock et au **getAttFromAttribute()** pour assigner les attributs du bloc cartouche via les méthodes privées **extract…()** et pour le dessiner. Notons que le libellé des voies et celui des indices de composition ne sera pas positionné lors du dessin du cartouche mais dans le programme principal, car le modèle adopté ne le permet pas. Par ailleurs, les libellés des BT et des RT ne seront pas dessinés par manque de place. Le centre de Tours devra changer les spécifications de dimension pour le cartouche.

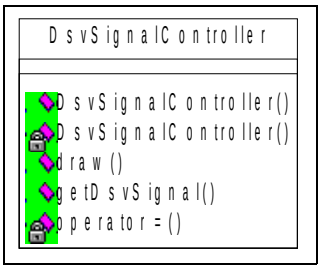
- **DsvVoieController**



Le constructeur prend une instance d'**Entity** pour initialiser le membre hérité : **_entity.** Elle définit une méthode **getDsvVoie()** qui utililise la méthode héritée **getEntity()** pour récupérer l'entité de type **DsvVoie** du mapping via un casting dynamique. Enfin la méthode **draw()** fait appel au DrawerBlock et au **getAttFromAttribute()** pour assigner les attributs du bloc voie. Toutes les informations concernant la voie sont obtenues grâce aux **getDsvVoie().** En effet **getDsvVoie()** renvoie un businessObject **DsvVoie** qui à travers **getLenth()** donne la longueur de la voie et à

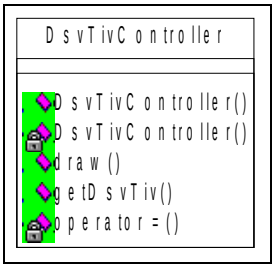
travers **getLb()** donne le libellé de la voie, ainsi **draw()** peut dessiner la voie avec les bonnes informations.

- **DsvSignalController**



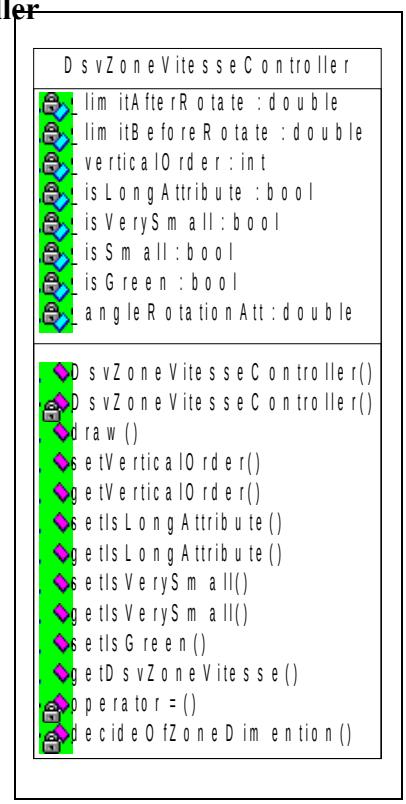
Le constructeur prend une instance d'**Entity** pour initialiser le membre hérité : **_entity.** Elle définit une méthode **getDsvSignal()** qui utilise la méthode héritée **getEntity()** pour récupérer l'entité de type **DsvSignal** du mapping via un casting dynamique.
Jusqu'à maintenant, la méthode **draw()** dessine un seul bloc associé à un businessObject. Dans cette classe, la méthode **draw()** gère deux blocs, un pour les signaux répétés et un pour les signaux non répétés car leur représentation graphique diffère. C'est regrettable de perdre la correspondance d'un identifiant de bloc pour un businessObject, mais les contraintes du DSV l'imposent. Ce choix rend impossible de changer la base de données Access à partir d'un objet dessiné sur AutoCAD dans une extension future de l'application.

- **DsvTivController**



Le constructeur prend une instance d'**Entity** pour initialiser le membre hérité : **_entity.** Elle définit une méthode **getDsvTiv()** qui utilise la méthode héritée **getEntity()** pour récupérer l'entité de type **DsvTiv** du mapping via un casting dynamique.
La même remarque est à faire que la classe précédente pour la méthode **draw()** car on aura deux blocs à gérer, un pour les Tiv fixe et un pour les Tiv mobiles.

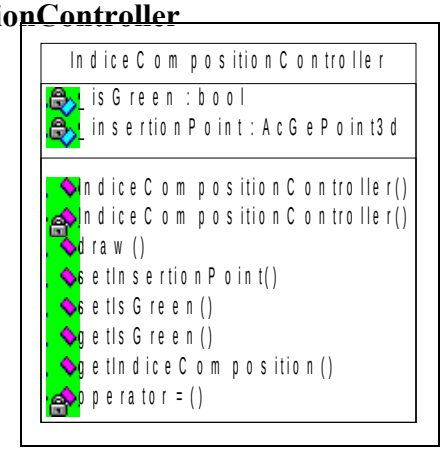
- **DsvZoneVitesseController**



Le constructeur prend une instance d'**Entity** pour initialiser le membre hérité : **_entity.** Elle définit une méthode **getDsvZoneVitesse()** qui utilise la méthode héritée **getEntity()** pour récupérer l'entité de type **DsvZoneVitesse** du mapping via un casting dynamique.

La méthode **draw()** est plus complexe que les précédentes. Non seulement elle aura à gérer plusieurs blocs, puisque les arrêtes d'une zone de vitesse changent de caractéristiques selon sa position, mais aussi elle aura à faire des traitements selon la dimension et la position de la zone. En d'autre terme une zone a besoin de connaître quelques caractéristiques de la zone qui la précède. Ainsi des booléens comme **_isVerySmall**, **_isSmall** renseignent sur la position de l'attribut vitesse de la zone, **_isLongAttribut** renseigne sur la forme que doit avoir l'attribut, **_verticalOrder** donne l'ordre vertical de la zone de vitesse, **_isGreen** spécifie la couleur de texte à utiliser (noir ou vert). Ces différentes variables ne peuvent être assignées ici, puisqu'une zone de vitesse ne connaît qu'elle même, ils seront donc assignés dans le composant logiciel permettant l'agencement de tous les objets graphiques du DSV, c'est à dire le composant **ToolDsvDrawer**.

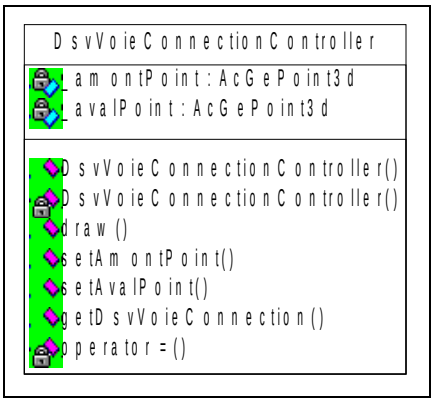
- **DsvIndiceComopositionController**



Le constructeur prend une instance d'**Entity** pour initialiser le membre hérité : **_entity.** Elle définit une méthode **getIndiceComposition()** qui utilise la méthode héritée **getEntity()** pour récupérer l'entité de type **IndiceComposition** du mapping via un casting dynamique.

La méthode **draw()** gère cette fois un seul bloc mais utilise le membre **_isGreen** pour spécifier la couleur du texte. Le businessObject associé à ce contrôleur ne possède pas d'information spécifiant un point d'insertion graphique, on a donc besoin d'un membre **_insertionPoint.**

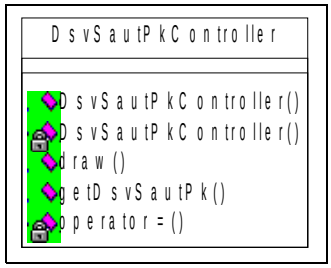
- **DsvVoieConnectionController**



Le constructeur prend une instance d'**Entity** pour initialiser le membre hérité : **_entity.** Elle définit une méthode **getDsvVoieConnection()** qui utilise la méthode héritée **getEntity()** pour récupérer l'entité de type **DsvVoieConnection** du mapping via un casting dynamique.

La méthode **draw()** aura deux blocs à gérer, un pour les connections montantes et un pour les connections descendantes. Le critère de décision pour dessiner tel ou tel bloc est la comparaison des ordonnées de **_amontPoint** et **_avalPoint**.

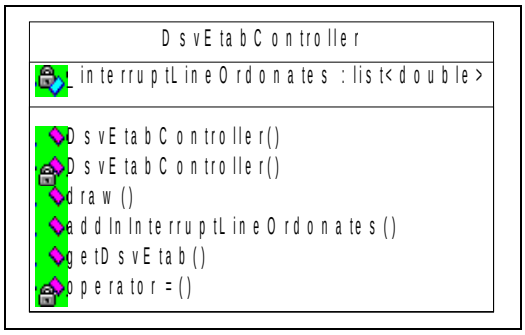
- **DsvSautPkController**



Le constructeur prend une instance d'**Entity** pour initialiser le membre hérité : **_entity.** Elle définit une méthode **getDsvSautPk()** qui utilise la méthode héritée **getEntity()** pour récupérer l'entité de type **DsvSautPk** du mapping via un casting dynamique.

La méthode **draw()** gère un seul bloc avec deux attributs qui sont les pk aval et amont, leur valeur est récupérée grâce à **getDsvSautPk()** qui permet de récupérer l'implantation du saut de pk.

- **DsvEtabController**



Le constructeur prend une instance d'**Entity** pour initialiser le membre hérité : **_entity.** Elle définit une méthode **getDsvEtab()** qui utilise la méthode héritée **getEntity()** pour récupérer l'entité de type **DsvPointSingulier** qui sont des établissements.

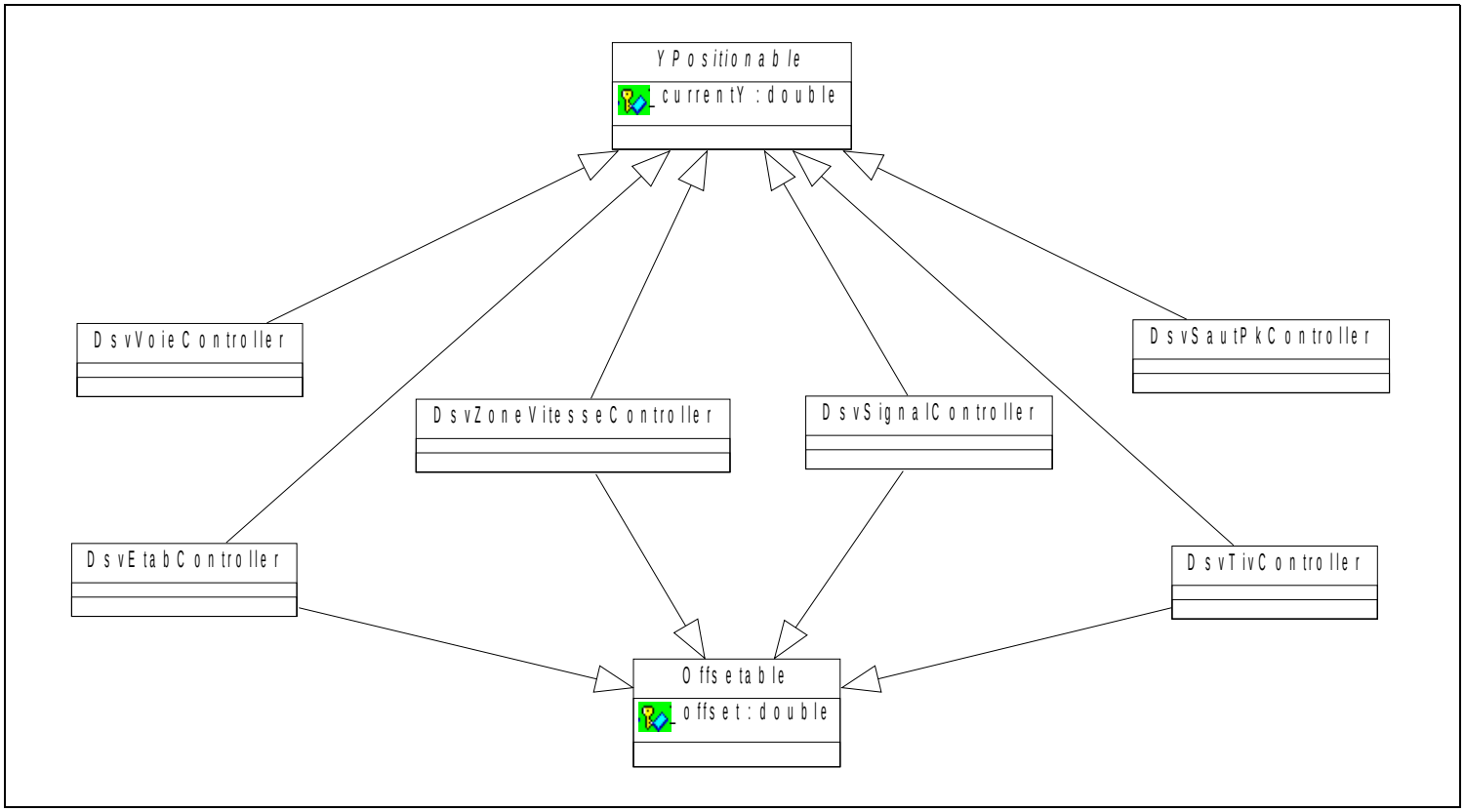
La méthode **draw()** gère deux blocs, un pour l'établissement et un pour le repérage en pointillés de l'établissement. Le membre **_interruptLineOrdonates** enregistre les ordonnées des points délimitant les pointillés. Ce membre ne peut être initialisé que dans le **ToolDsvDrawer** avec la méthode **addInInterruptLineOrdonates()** qui est public.

Pour tous les contrôleurs, on a vu qu'on définissait une méthode **draw()** pour dessiner les blocs AutoCAD. Une question se pose, d'où viennent les coordonnées des points d'insertion ?
C'est les classes de spécification qui fournissent la solution.

- **Les classes de spécification**

Prenons par exemple le cas des signaux. Le point origine du DSV est le point en bas à gauche du cartouche. Un signal est repéré par sa distance par rapport à la voie qui le contient. Cette distance est obtenue de la façon suivante : **getDsvSignal()** renvoie le businessObject **DsvSignal** à partir duquel on extrait la distance avec la méthode **getDistance().** Pour repérer le signal dans tout le DSV on a besoin de la distance de début de voie et du décalage par rapport à l'origine. Le décalage par rapport à l'origine est donné par **_horizontalOffset** du **ControllerBlock.**
Mais la distance de début de voie n'est pas connue par le signal. Ce problème est commun aussi aux Tiv, aux zones de vitesse et aux établissements car tous sont liés à la voie qui les contient.
La solution choisi pour résoudre ce problème est l'héritage multiple. Les contrôleurs **DsvSignalController**, **DsvTivController**, **DsvZoneVitesseController** et **DsvEtabControllerController** héritent d'une classe **Offsetable** qui encapsule un membre **_offset** donnant la position relative des voies.

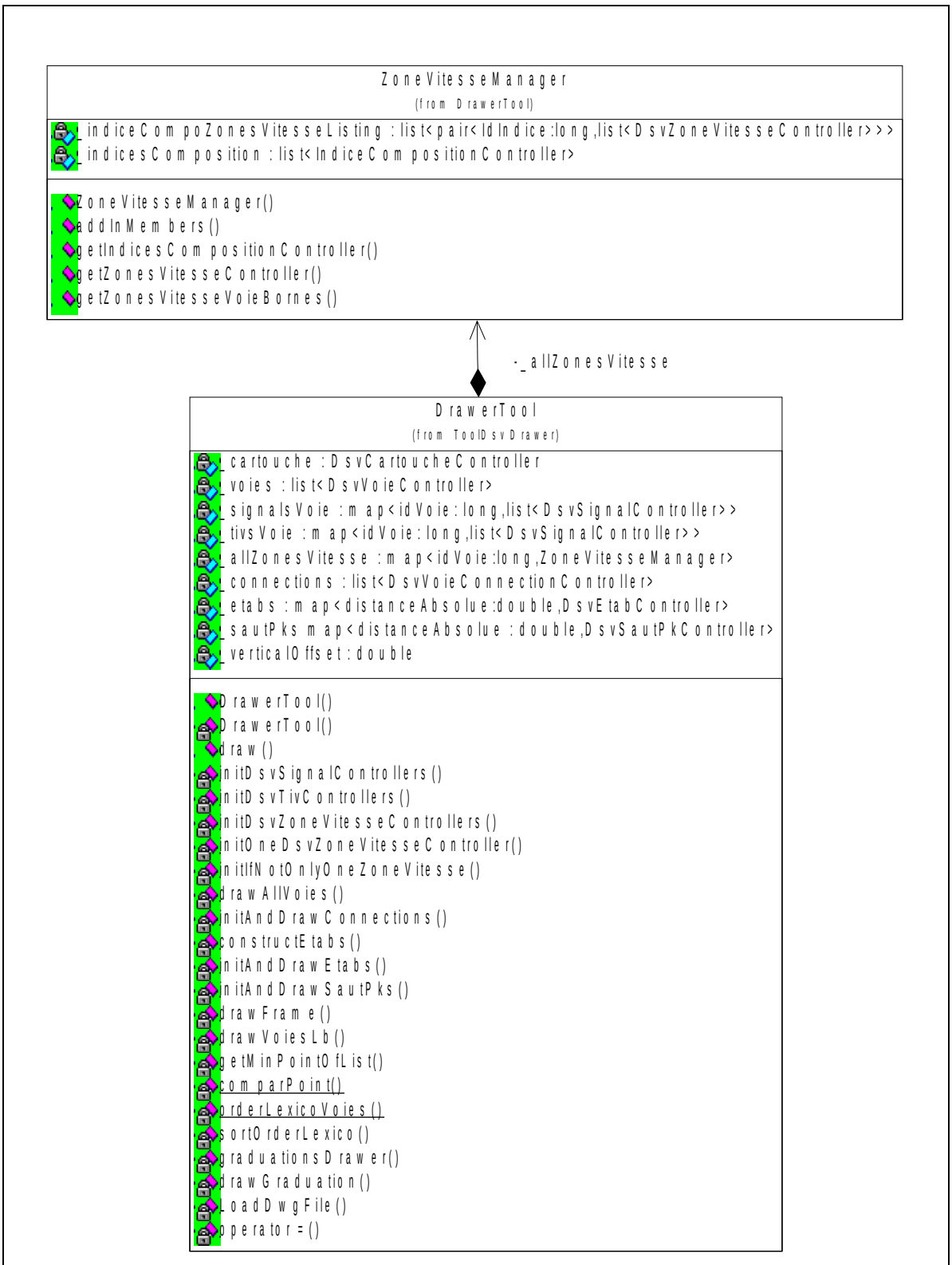
Une autre information manque dans les contrôleurs. Si d'après ce qui précède, on connaît l'abscisse d'un signal, il reste à avoir comment faire pour récupérer son ordonnée, surtout que le modèle de données ne prévoit pas l'agencement des différents objets du DSV. La solution retenue est encore l'héritage multiple. Une classe de spécification **Ypositionnable** encapsule un membre **_currentY** donnant l'ordonnée d'insertion pour **DsvVoieController, DsvSignalController, DsvTivController DsvZoneVitesseController, DsvEtabControllerController** et **DsvSautPkController.**



L'initialisation de **_currentY** et de **_offset** se fait à l'aide d'un accèsseur dans le composant chargé de l'agencement des objets graphiques, c'est à dire le **ToolDsvDrawer**. Cette initialisation demande une certaine intelligence que seul le **ToolDsvDrawer** possède.

ToolDsvDrawer

Ce composant logicielle possède une seul classe **DrawerTool** qui a une vue globale des **ObjectsController** et donc aussi des **BusinessObject**.



La classe **DrawerTool** agrège une classe interne, **ZoneVitesseManager,** qui encapsule un bloc de zones de vitesse associées à une voie. La description de cette classe interne se fera lors de la description du membre **_allZonesVitesse** de **DrawerTool**.

Le constructeur de **DrawerTool** prend en paramètre une **Entity** Dsv avec laquelle il initialise le membre **_cartouche** qui est un **DsvCartoucheContoller**. Il initialise ensuite le membre **_verticalOffset** qui définit les décalages verticaux constants entre les objets du Dsv.

Il appelle ensuite **LoadDwgFile()** qui charge les blocs AutoCAD en appelant le **LoaderBlock** d'**AcadUtil**. Une série d'initialisation de certains membres de la classe est faite dans le constructeur à travers des fonctions membres privées de **DrawerTool.**

- **_voies**

Ce membre est un conteneur C++. C'est à dire une liste de **DsvVoieController**. Pour l'initialiser, on a besoin d'une liste d'**Entity** de BusinessObject dont le type « caché » est **DsvVoie**. On obtient cette liste en utilisant les fonctions membres de **_cartouche**. En effet, le membre **_cartouche** a accès à **Dsv** de **BusinessObject** et donc à la méthode **loadVoies()**.

- **_signalsVoie**

Ce membre est une map au sens C++. C'est un ensemble de listes de **DsvSignalController** trié avec une clé qui est l'identifiant de la voie. Ainsi on a facilement accès à la liste des contrôleurs de signaux d'une certaine voie. Ce membre est initialisé à travers la fonction privée **initDsvSignalControllers()** qui charge les signaux d'une voie de la base à travers les membres de **_voies.**

- **_tivsVoie**

Ce membre est une map au sens C++. C'est un ensemble de listes de **DsvTivController** trié avec une clés qui est l'identifiant de la voie. Ainsi on a facilement accès à la liste des contrôleurs de tableau indicateur de vitesse d'une certaine voie. Ce membre est initialisé à travers la fonction privée **initDsvTivControllers()** qui charge les Tiv d'une voie de la base à travers les membres de **_voies.**

- **_allZonesVitesse**

Ce membre est une map au sens C++. C'est un ensemble de **ZoneVitesseManager** trié avec une clé qui est l'identifiant de la voie. A chaque voie, correspond un et un seul **ZoneVitesseManager** qui gère un ensemble de blocs de zones de vitesse.

  - **ZoneVitesseManager**

  Cette classe interne contient deux variables membres et quelques fonctions membres utilitaires :

    - **_indiceCompoZonesVitesseListing**

    C'est une liste de paire d'identifiant d'indice de composition et de liste de **DsvZoneVitesseController**. Le choix de cette structure est dû au fait qu'à une voie, est associé un ensemble d'indice de composition (un ensemble de catégorie de train), et à chaque indice de composition, on a un ensemble de zones de vitesse.

    - **_indicesComposition**

    C'est une liste contenant l'ensemble des **IndiceCompositionController** d'une voie.

    - **addInMembers()**

    Cette fonction permet l'initialisation les deux membres précèdants. Elle prend comme paramètre un **IndiceCompositionController** et une liste de **DsvZoneVitesseController** bien initialisés.

    - **getZonesVitesseController()**

    Prend en paramètre un identifiant d'indice de composition et renvoie une liste de **DsvZoneVitesseController.**

    - **getZonesVitesseVoieBornes()**

    Cette fonction renvoie une paire de double qui sont les ordonnées délimitant un bloc de zones de vitesse. Ceci va nous servir pour dessiner les pointillés separant les établissements.

L'initialisation de **_allZonesVitesse** se fait à travers la fonction **initDsvZoneVitesseControllers()** qui gère tous les cas de zones possibles.

- **_connections**

Ce membre est une liste de **DsvVoieConnectionController**. Les contrôleurs des connections sont initialisés et dessinés à travers **initAndDrawConnections()** qui utilise le membre **_allZonesVitesse** pour repérer l'implantation des connections.

- **_etabs**

Ce membre est une map au sens C++. C'est un ensemble de **DsvEtabController** trié avec une clé qui est la distance absolue (par rapport à l'établissement d'origine) d'implantation de l'établissement. Les contrôleurs des établissements sont initialisés et dessinés à travers **constructEtabs()** et **initAndDrawEtabs()**. La première fonction sert à construire les contrôleurs avec les bonnes « entities » et le second fait appel à **getVitesseVoieBornes()** pour le dessin des pointillés.

- **_sautPks**

Ce membre est une map au sens C++. C'est un ensemble de **DsvSautPkController** trié avec une clé qui est la distance absolue (par rapport à l'établissement d'origine) d'implantation de l'établissement. Les sauts de pk sont initialisés et dessinés à travers **initAndDrawSautPks()**.

La fonction membre **drawAllVoies()** dessine toutes les voies, avec les signaux, les Tiv et les zones de vitesse.

- **La fonction drawAllVoies()**

Le DSV impose un ordre de dessin spécifique pour les voies. Avant de dessiner les voies, on ordonne la liste **_voie** suivant la fonction **sortOrderLexico ()**. Une voie V1 est plus petite que V2 si et seulement si la DistanceOffset de V1 est plus petite que la DistanceOffset de V2, s'il y a égalité, V1 est plus petite que V2 si et seulement si la longueur de V1 est plus grande que V2. Une fois l'ordre établit, un algorithme de placement permet de positionner les voies l'une par rapport à l'autre. Pour l'agencement, on parcourt la liste des **DsvVoieController** triés et on place les voies sur le même niveau si possible, sinon on passe au niveau suivant. A chaque dessin d'un **DsvVoieController,** on dessine ses **DsvSignalController**, ainsi que ses **DsvTivController** et ses **DsvZoneVitesseController**.
Cette fonction gère plusieurs variables locales pour réaliser le placement des objets précédemment cités du DSV.

La fonction **draw()** est charger de dessiner tout le DSV. Elle appelle :

- ✓ Le **draw()** de **_cartouche**.
- ✓ Le **drawAllVoies()**.
- ✓ Le **initAndDrawConnections()**.
- ✓ Le **initAndDrawEtabs()**.
- ✓ Le **initAndDrawSautPks()**.

Elle appelle par ailleurs les fonctions : **drawFrame()**, **drawVoiesLb()**, **graduationsDrawer()**.

- **Les fonctions drawFrame() et drawVoiesLb()**

La première dessine le cadre général du Dsv sans utilisation du contrôleur, et la deuxième écrit sur le cartouche du DSV les libellés des voies.

- **La fonction graduationsDrawer()**

Lors de l'analyse du cahier des charges, on avait précisé que les graduations ne correspondaient à un BusinessObject du modèle. Donc leur dessin se fera sans contrôleur.
La difficulté du dessin de la graduation réside dans sa non régularité. En effet, dés qu'il y a un saut de pk, il y une discontinuité de la graduation. De plus, sur un intervalle entre deux sauts de pk ou bien entre l'établissement de départ et le premier saut de pk ou encore entre le dernier saut de pk et

l'établissement de fins, le pk peut être croissant comme décroissant. Par ailleurs on doit représenter les valeurs de pk sur les graduations par pas de cinq.
La solution adoptée est la suivant :
On se place sur le premier saut de pk, celui qui a la distance d'implantation la plus petite. On compare son pk amont avec le pk de l'établissement d'origine, on obtient donc l'information de croissance ou de décroissance. Selon cette information, on calcule la partie entière inférieure ou supérieure du pk amont. Cette dernière information, ainsi que l'information de croissance, la distance du saut de pk et celle de l'établissement d'origine sont fournis en paramètre à la fonction **drawGraduation().** On fait de même pour le deuxième saut de pk en changeant l'établissement d'origine avec le premier saut de pk et en prenant son pk aval. Pour la fin de la graduation, on prend le pk de l'établissement de destination et le pk aval du dernier saut de pk.

- **La fonction drawGraduation()**

A partir des informations qu'on lui fournit, elle dessine des blocs AutoCAD représentant une seule graduation, chacun en ajustant leur échelle verticale à l'aide de compteurs arithmétiques. Elle permet aussi le dessin de la valeur du pk à l'aide de compteurs.

Ainsi, la classe **DrawerTool** fournit une méthode **draw()** qui dessine le DSV passé en paramètre au constructeur de cette classe.

Nous allons voir maintenant comment se fait l'interfaçage avec AutoCAD ainsi que l'architecture de l'IHM.

DsvDrawerArxMain

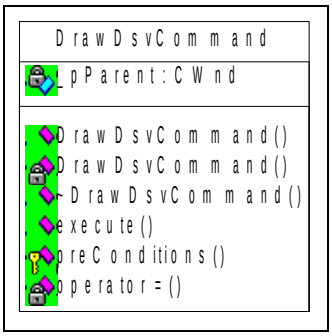
Ce composant contient deux classes : **DrawDsvCommand** et **DsvListDialog**, et un module de fonctions déclarées et définies dans un fichier **DsvDrawerArxMain**. La description de ce dernier aboutit à la description des deux classes précédentes.

- **Les fonctions du fichier DsvDrawerArxMain**

Le choix de faire un fichier de fonctions non encapsulées dans une classe objet est dû au fait que ce fichier est un exécutable qui va être chargé par AutoCAD sous d'ARX (DLL AutoCAD).

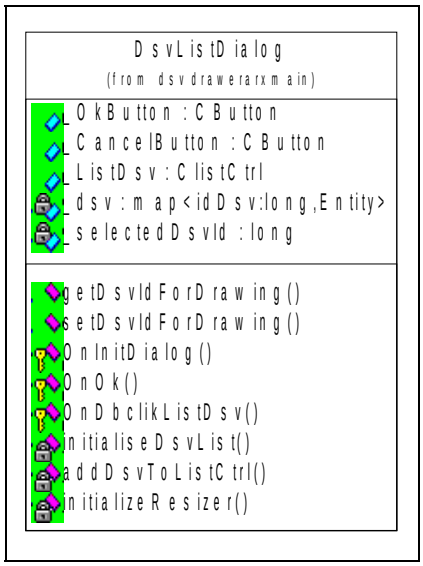
- **initApp()**
  Définie le nom de la commande dessinerDsv à taper sur AutoCAD et la relie à la fonction définissant la commande. En l'occurrence **dessinerDsv().**
- **unloadApp()**
  Inhibe la commande dessinerDsv.
- **dessinerDsv()**
  Cette fonction récupère un handel sur la fenêtre AutoCAD, définie une instance *acadlog* de **AcadLogEventSubscriber (**Voire **AcadUtil)**, puis instancie un **DrawerDsvCommand** en lui passant en paramètre le handel précédant et *acadlog.*
- **DllMain()**
  Cette fonction attache la DLL à AutoCAD.
- **acrxEntryPoint()**
  Cette fonction est appelée automatiquement juste après la **DllMain()** et suivant les messages reçus par la fenêtre AutoCAD. Elle exécute : **initApp(), unloadApp()** ou ferme la base de données Access ouverte par le **DrawerDsvCommand.**

- **La classe DrawDsvCommand**

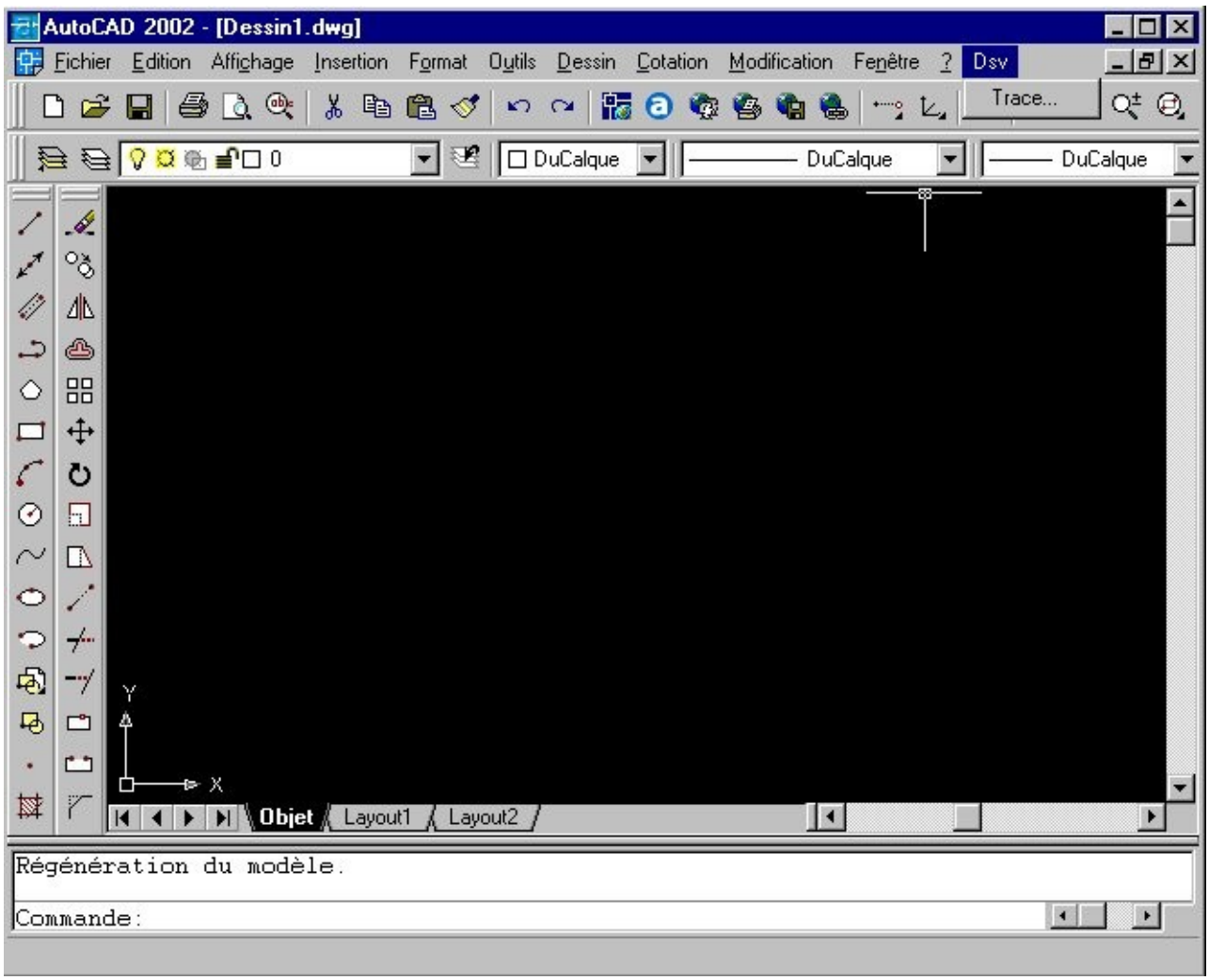


Le constructeur prend en paramètre une fenêtre windows et un **AcadLogEventSubscriber** pour informer l'utilisateur par le biais de messages sur la fenêtre AutoCAD.
Cette classe permet de gérer les interactions entre l'utilisateur et les boîtes de dialogue.
La fonction **preCondition()** permet de gérer la connexion à la base Access en utilisant le composant **DbLogin.**
La fonction **execute()** charge tous les DSV existant en base et instancie **DsvListDialog** qui présentera la liste des DSV à l'utilisateur. Elle récupère l'identifiant du DSV sélectionné et appelle le **draw()** du **DrawerTool** avec le bon DSV.

- **La classe DsvListDialog**



Cette classe hérite de **Cdialog**, la classe de base MFC gérant les boites de dialogue.
Son constructeur prend une fenêtre et une map d' **Entity,** de type « caché » **Dsv,** dont la clé est l'identifiant de l' **Entity**. Elle initialise la boîte de dialogue à travers **OnInitDialog()** pour présenter la liste des **Dsv.**
Cette classe fournit à la classe **DrawDsvCommand** l'identifiant du DSV à dessiner.
Notons que les boîtes de dialogue utilisées sont des boîtes « modales », c'est à dire, elles monopolisent le processeur par rapport à l'application mère qui les lance.
Ainsi pour récapituler, voilà ce que le **DsvDrawerArxMain** est capable de faire **:**
L'utilisateur lance AutoCAD :



Fenêtre AutoCAD

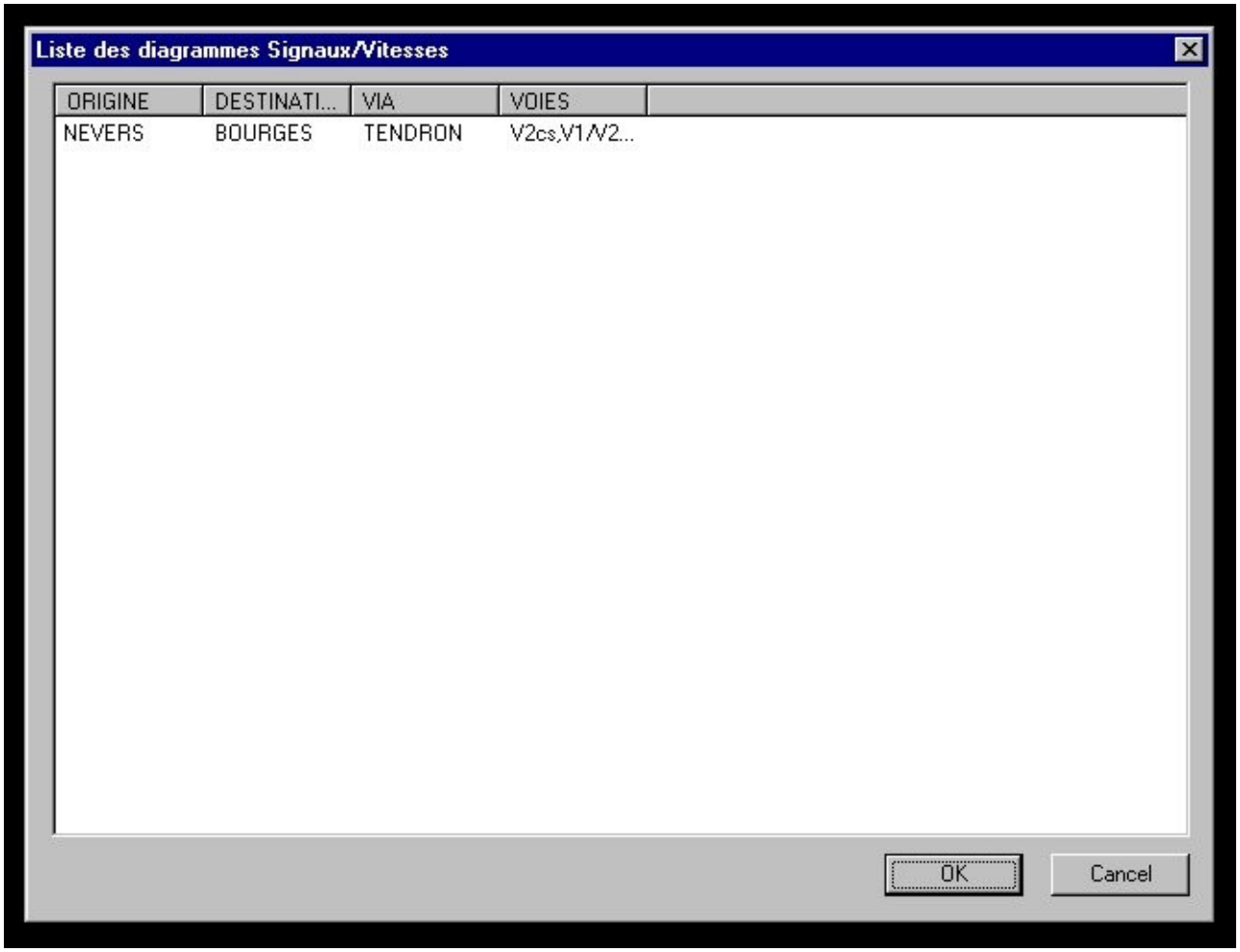
Il a ensuite deux possibilités : taper sur la ligne de commande dessinerDsv ou bien utiliser l'onglet Dsv en haut à droite qui fournit une fonction Tracer. Dans un fichier de configuration AutoCAD on a lié la fonction Tracer à la commande dessinerDsv.

Après click sur Tracer la boite de connection à la base de donnée est présentée à l'utilisateur. La gestion de login et de mot de passe n'est pas encore réalisée. On choisit la base Access contenant les
DSV.

Boîte de connexion

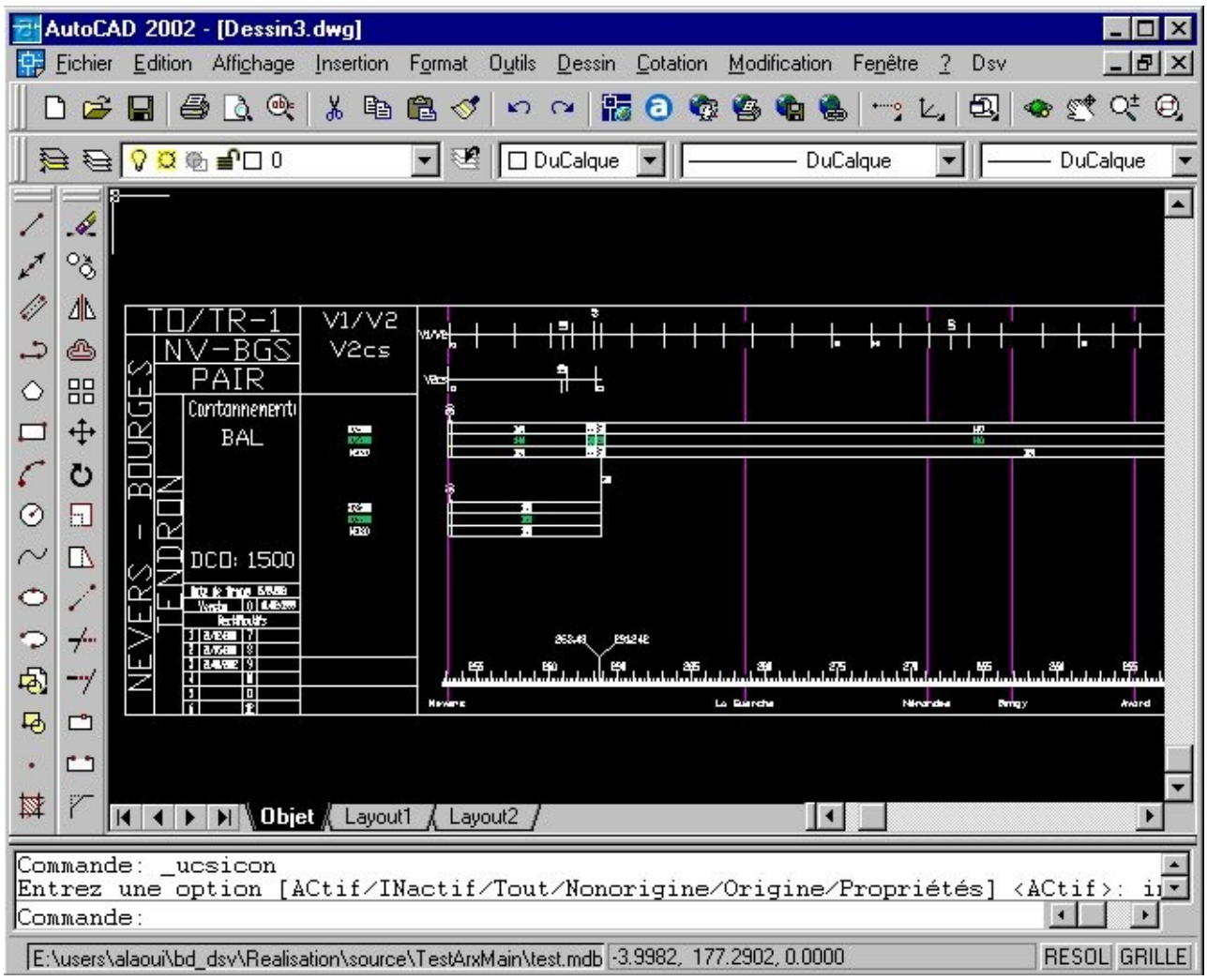
Base de données Access

Une boîte de dialogue s'affiche ensuite avec la liste des DSV. Pour l'instant il n'existe qu'un seul tronçon de DSV en base : le Nevers-Bourges.



Liste des DSV

L'utilisateur choisit le DSV en double cliquant dessus ou bien en le sélectionnant et appuyant sur OK. Le DSV apparaît alors sur la fenêtre AutoCAD.



DSV Nevers-Bourges

Résultats

Les résultats du dessin obtenu sont comparés au calque du DSV qui a été envoyé par le centre de vérification des DSV. La correspondance des DSV a permis de valider le prototype de l'application dessinant les DSV.

Une réunion avec les responsables du centre de vérification c'est faite le 11 septembre. A l'issue de cette réunion certains points concernant les DSV ont été mis au claire et doivent être pris en charge par l'extension future de l'application. Nous allons examiner ces points dans ce qui suit.

- **Voies :**

La cellule qui mentionne les voies représentées sur le DSV comporte au maximum 3 niveaux.

Si le DSV représente plus de 3 niveaux de voies, seules les voies les plus importantes, c'est à dire celles qui présentent les tronçons les plus longs sur le DSV seront mentionnées sur le cartouche.

Si une même voie change de nom, ses différents noms sont mentionnés sur la même ligne et séparés par « / ». Les différents noms d'une voie seront mentionnés de façon automatique uniquement si elle a été constituée à partir de plusieurs voies qui portent des noms différents dans les BT dont elles sont issues.

La police utilisée pour les noms de voie est jugée trop grande.

- **Cantonnement :**

Un DSV peut comporter plusieurs indications de cantonnement et donc le modèle de données et à retoucher.

- **DCO :**

Un DSV peut comporter plusieurs indications de DCO et donc le modèle de données et à retoucher.

La police choisie dans l'outil d'édition est jugée trop grande.

- **Indices de composition :**

La police choisie dans l'outil d'édition est jugée trop petite.

Les différents indices de composition sont superposés, présentés du matériel le plus rapide (haut) au matériel le plus lent (bas).

Lors de l'édition d'un DSV, un panneau de dialogue permettra à l'utilisateur de choisir les indices de composition qu'il souhaite voir représentés.

Les indices de composition ne seront plus présentés dans le cartouche mais dans la partie principale du DSV.
Pour permettre l'affichage des indices de composition en partie centrale, les lignes de signaux, de limitations de vitesse et des pks seront décalées de 25mm sur la droite.
En effet, par souci de lisibilité, les indices de composition seront directement accolés aux lignes de limitations de vitesse auxquelles ils correspondent.
Chaque indice de composition sera mentionnés à gauche de la première cellule de « sa » ligne de limitation de vitesse, et répété à droite de la dernière cellule de cette ligne.
Ce choix sera intégré dans la prochaine version du référentiel d'élaboration des DSV.

Les indices de composition qui présentent les mêmes limitations de vitesse seront présentés sur la même ligne et séparés par « / ». L'accolement de plusieurs indices de composition pourra nécessiter un décalage supplémentaire des lignes de signaux, de limitation de vitesse et de PR vers la droite.

- **BT et RT :**

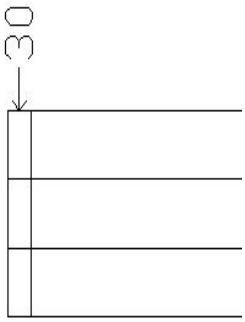
Tous les BT et RT utilisés dans la construction d'un DSV doivent être mentionnés dans son cartouche.
Si le nombre de BT et/ou de RT à mentionner est/sont important(s), les cases correspondantes peuvent être agrandies jusqu'à la ligne qui porte la mention de la date de tirage. Le nombre d'indices de composition qui peut être affiché sera dans ce cas limité. Si le nombre d'indices de composition à représenter est important, il sera nécessaire d'éditer plusieurs DSV.
La police ARIAL TT 5mm sera conservée quel que soit le nombre de RT et de BT à mentionner

- **Valeur des vitesses dans les zones de vitesse**

La police choisie dans l'outil d'édition est jugée trop petite.

Quand la longueur de la limitation de vitesse l'autorise, la valeur de la vitesse limite est inscrite horizontalement dans la cellule correspondante.
Quand la cellule est trop étroite, l'outil tente d'inscrire la valeur de la vitesse limite verticalement dans la cellule**.**
Quand la cellule est trop petite pour porter l'indication de vitesse verticalement, la vitesse limite est mentionnée au dessus de la zone de limitation de vitesse. Une flèche verticale dirigée vers le bas pointe sur la cellule correspondante.

Exemple :

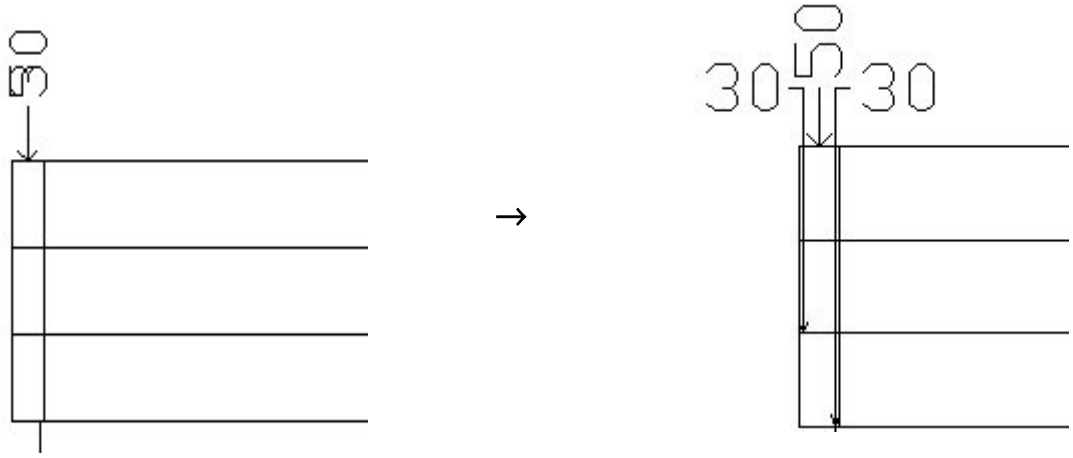


Lorsque plusieurs cellules trop étroites pour contenir des indications de vitesse sont superposées, l'outil d'édition, dans sa version actuelle, superpose au dessus de ces cellules la valeur des vitesses limites portées par chacune d'elles.
Ce système ne permet pas de mentionner de façon lisible des limites de vitesse différentes pour différents indices de composition.
Dans ce cas, rare, plusieurs alternatives sont envisagées :

- Mentionner les différentes limites de vitesse côte à côte au dessus des cellules superposées et assurer la lisibilité de l'ensemble grâce à des flèches indiquant les valeurs limites affectées à chaque indice de composition.

Cette solution ne peut pas être automatisée. Elle demandera l'intervention de l'utilisateur pour déplacer les valeurs des limites de vitesse et adapter les flèches.

Exemple :



- Présentant les cellules très étroites et comportant des limitations de vitesse différentes sur la même portion de voie pour différents indices de composition sur 2 DSV différents.

- **Connexion entre voies**

Les connexions entre voies du DSV sont figurées par des flèches qui représentent les mouvements possibles d'une voie sur l'autre dans le sens de circulation dépouillé.
Ce mode de représentation peut s'avérer illisible dans le cas d'une trop grande densité de connexions à représenter (passages d'aiguilles rapprochés).

Pour résoudre ce problème, un panneau de dialogue permettra à l'utilisateur, lors de l'édition, de choisir les connexions qu'il souhaite afficher. Par défaut, toutes les connexions seront affichées.

- **Graduation des Pk**

Le respect des arrêts dans les gares d'arrêt obligatoire n'est actuellement pas contrôlé dans le cadre des vérifications complètes.
Il n'est donc pas prévu de repérer sur les DSV les gares d'arrêt obligatoire.

PARTIE N°5.CONCLUSION

Les outils nécessaires pour permettre une édition flexible des DSV n'ont pas encore été développés. L'utilisateur de l'application ne pourra pas, par exemple, choisir certaine voies d'un DSV pour n'éditer que celles-ci. Cependant un prototype éditant tout le DSV est fonctionnel. Le but du stage était précisément la réalisation de ce prototype. Par ailleurs, l'application actuelle fournit des classes réutilisables permettant l'extension du prototype.

Outre le contrôle et la vérification des fichiers ATTESS, les DSV peuvent servir de support informatique pour retracer automatiquement l'itinéraire des trains munis du système ATTESS. Il est prévu qu'à partir d'une mission ATTESS et d'un réseaux de neurones on puisse affirmer le train effectuant cette mission a traversé telle ou telle voie.

PARTIE N°6.GLOSSAIRE

Définition des abréviations utilisées :

GDA : Génie Décisionnel Appliqué, unité du service de la direction de la recherche SNCF**.**

ATESS : Acquisition et Traitement des Avènement de Sécurité en Statique.

DSV : Diagrammes Signaux/Vitesses.

IHM : Interface Homme Machine.

MFC : Microsoft Fondation Classe.

LPB : Long Parcours Binaire.

BT : Bande Type.

RT : Renseignement Technique.

DAO : Data Access Object.

DLL : Data Link Library.

PK : Point Kilométrique.

PARTIE N°7.BIBLIOGRAPHIE

[SM02] Scott Meyers., “Effective C++ Second Edition 50 specific ways to improve your programs and designs”.