Titre: Title:	Contribution à un système d'exploitation générique d'infrastructures urbaines avec composantes mobiles
Auteur: Author:	Dougoukolo Konaré
Date:	2001
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Konaré, D. (2001). Contribution à un système d'exploitation générique d'infrastructures urbaines avec composantes mobiles [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6982/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6982/
Directeurs de recherche: Advisors:	Samuel Pierre
Programme: Program:	Génie électrique

UNIVERSITÉ DE MONTRÉAL

CONTRIBUTION À UN SYSTÈME D'EXPLOITATION GÉNÉRIQUE D'INFRASTRUCTURES URBAINES AVEC COMPOSANTES MOBILES

DOUGOUKOLO KONARÉ
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
FÉVRIER 2001

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

CONTRIBUTION À UN SYSTÈME D'EXPLOITATION GÉNÉRIQUE D'INFRASTRUCTURES URBAINES AVEC COMPOSANTES MOBILES

présenté par : KONARÉ Dougoukolo

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

Mme. BELLAÏCHE Martine, M.Sc., président

M. PIERRE Samuel, Ph.D., membre et directeur de recherche

M. GUIBAULT François, Ph.D., membre

À mes parents...

REMERCIEMENTS

Je tiens à remercier particulièrement mon directeur de recherche, Monsieur Samuel Pierre, qui m'a apporté tout son soutien tout au long de ce travail de recherche. Je suis très reconnaissant de la confiance qu'il a porté en moi et le remercie sincèrement du fond du cœur.

Je dois un grand merci à Guy Leclerc, Yves Dion, Luca Rossi ainsi que tous les membres du SIGEC et du LARIM qui ont su apporter une ambiance de travail chaleureuse dans un contexte de recherche universitaire. Les discussions que nous avons eues ont su enrichir nos connaissances tant au niveau académique que culturel.

Mes remerciements s'adressent également aux différents services de l'hôtel de ville à la Ville de Verdun et particulièrement à France Bernard.

Un grand merci à mes parents à ma famille ainsi qu'à mes amis.

RÉSUMÉ

À l'heure du développement durable, la réhabilitation des infrastructures urbaines reste la principale préoccupation des municipalités en Amérique du Nord. La gestion des systèmes urbains est et a toujours été le fruit de la collaboration de plusieurs acteurs. Elle consiste en la planification, la réhabilitation, la maintenance et la construction de nouveaux éléments, ainsi que la gestion en temps réel, la localisation géographique des éléments d'inventaires, la gestion des plaintes et des employés.

Ces infrastructures urbaines sont soumises à des contraintes de plus en plus sévères dues au transfert des responsabilités et aux diminutions des ressources tant au niveau financier qu'humain. Le manque de communication entre départements cause la duplication, voir l'incohérence au niveau des données et de l'information manipulée au sein de l'organisation. Le manque d'intégration au niveau des services, des outils logiciels et des données respectifs à chaque infrastructure n'améliore pas non plus la gestion. Aujourd'hui, seuls une concertation entre les différents groupes et un suivi continu des infrastructures urbaines permettront de réaliser une synergie qui façonnera la réhabilitation de celles-ci.

Dans ce contexte, la définition de Systèmes Intégrés D'EXploitation (SIDEX) dédiés aux diverses infrastructures urbaines apparaît comme une solution viable aux différents problèmes rencontrés par les municipalités. Un SIDEX se compose de plusieurs modules chacun en charge de la gestion d'une partie de l'infrastructure. Les modules du SIDEX se composent de données permanentes structurées dans une base de données, de paquetages formant la logique de l'application et des interfaces. Les concepts orientés objet ont été systématiquement utilisés durant le processus de conception des SIDEX, ce qui a permis un développement modulaire. Pour assurer une meilleure réutilisation au niveau de la conception, un SIDEX générique a été élaboré. En effet, les infrastructures qui sont à l'étude ont toutes des points en commun qui méritent

d'être regroupés en un système représentatif de chacune des infrastructures, d'où le qualificatif de générique.

Ce présent mémoire traite de la conception d'un SIDEX générique permettant aux municipalités de bénéficier d'un modèle conceptuel de données générique, et de développer des applications dédiées à la gestion des infrastructures urbaines. Ceci constitue un apport incontestable dans la perspective d'éventuels travaux de normalisation dont il est de plus en plus question dans ce domaine. En effet, plutôt que de concevoir chaque SIDEX de manière indépendante, il est préférable de concevoir un modèle conceptuel de base qui tient compte de tous les éléments communs aux différents systèmes urbains. De plus, le développement du SIDEX générique a pris en compte les aspects dynamiques de tous les systèmes urbains. La gestion en temps réel préconisée dans ce mémoire repose essentiellement sur les développements récents dans les domaines des télécommunications mobiles et fixes. Elle contribuera certainement à améliorer les processus de conservation et de réhabilitation des infrastructures urbaines.

ABSTRACT

Urban system management has always been the main concern of municipalities in North America. Urban system management consists of planning, rehabilitation, maintenance and construction of new elements; real-time management with monitoring devices; geographic location of inventory elements; management of complaints; management of employees and their different organization.

These urban infrastructures are subjected to increasingly severe constraints due to the transfer of responsibilities and the reductions of resources at the financial and human level. The management of urban infrastructures is not improved due to a lack of communication and interaction between services but also because of data duplication and incoherence between them. More integration between those services, software and their data would enhance management role. Moreover,. Nowadays only a dialogue between the various groups and a continuous follow-up will make a possible rehabilitation of those urban infrastructures.

In that way, an integrated operating system such as SIDEX, dedicated to the management, regulation and interactive and dynamic monitoring of each urban infrastructure would help municipalities in an efficient and effective manner. SIDEX is made of different packages, each in charge of several general management tasks. SIDEX package is comprised of structured and permanent data stored in database tables at the first level, classes representing business logic at the second level and interface templates at the third level. The use of object oriented concepts during the design process of SIDEX is highly recommended thus, allowing a distributed development of packages and interaction between them. To maximize reusability between SIDEX, a generic SIDEX should initially be design. All urban systems that have been studied have common features that should be grouped into a representative generic system, managed by a generic SIDEX. For example, the sewer system and the water system are very

similar, thus the design of a generic SIDEX will use the power of object-oriented language to instantiate SIDEX for each urban infrastructure.

With the design of a generic SIDEX in this thesis, municipalities will benefit a generic conceptual data model. This will be a real contribution in the standardization of a data model for future work in this field. Indeed, rather than conceiving each SIDEX in an independent way, it is preferable to design a basic conceptual model which takes account all the common elements of various urban systems. Moreover, the development of the generic SIDEX took into account the dynamic aspects of all urban systems. Management in real time, recommended in this thesis rests primarily on the recent developments in the fields of mobile and fixed telecommunications. It will certainly contribute to improve the processes of conservation and rehabilitation of urban infrastructures.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES SIGLES ET ABBRÉVIATIONS

API	Application Programming Interface
ASN.1	Abstract Syntax Notation number One
ASP	Active Server Pages
B2C	Business to Consumer
CASE	Computer Aided Software Engineering
CERIU	Centre d'Expertise et de Recherche en Infrastructures Urbaines
CGI	Common Gateway Interface
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheet
DAL	Data Abstraction Layer
DCOM	Distributed COM
DHTML	Dynamic HyperText Markup Language
DOM	Document Object Model
ECMAScript	European Computer Manufacturers Association Script
ESRI	Environmental Systems Research Institute
GSM	Global System for Mobile Communications
HDML	Handheld Device Markup Language
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IIOP	Internet Inter-ORB Protocol
IIS	Internet Information Server
IP	Internet Protocol
ISAPI	Internet Services API
JDBC	Java DataBase Connectivity

JSP	Java Server Pages
NSAPI	Netscape Services API
ODBC	Open DataBase Connectivity
OLE-DB	Object Linking and Embedding DataBase
OMG	Object Management Group
OMT	Object Modeling Technique
ORB	Object Request Broker
OSI	Open System Interconnection
PHP	Personal Home Pages hypertext preprocessor
PL/SQL	Procedural Language / Structured Query Language
PSTN	Public Switched Telephone Network
RAD	Rapid Application Development
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SGBD	Système de Gestion de Bases de Données
SIAD	Système Intelligent d'Aide à la Décision
SIDEX	Système Intégré D'EXploitation
SIG	Système d'Information Géographique
SIGEC	Système Intégré de GEstion Coordonnée
SIRS	Système d'Information à Références Spatiales
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP	Transport Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UML	Unified Modeling Language
VSA	Association Suisse des Professionnels de l'Épuration des Eaux
W3C	World Wide Web Consortium

WAP	Wireless Application Protocol
WDP	Wireless Datagram Protocol
WML	Wireless markup Language
WML	Wireless Markup Language
WSP	Wireless Session Protocol
WTLS	Wireless Transport Layer Security
XML	eXtensible Markup Language

CHAPITRE I
INTRODUCTION

À l'heure du développement durable, la réhabilitation des infrastructures urbaines reste une des principales préoccupations des municipalités en Amérique du Nord. D'un autre côté, la gestion des systèmes urbains est et a toujours été le fruit de la collaboration de plusieurs acteurs. Malheureusement, ces infrastructures sont soumises à des contraintes de plus en plus sévères en raison du transfert des responsabilités et de la diminution des ressources tant au niveau financier qu'humain. Aujourd'hui, seuls une concertation entre ces différents acteurs et un suivi continu des infrastructures urbaines permettront de réaliser une synergie qui façonnera leur réhabilitation et permettra ainsi de léguer ce patrimoine aux générations futures. Ce présent mémoire traite de la conception d'un Système Intégré D'Exploitation (SIDEX) générique pour applications urbaines mobiles multimédias. Parallèlement à ces travaux, un Système Intégré de GEstion Coordonnée (SIGEC), où seront intégrés les SIDEX, est en cours de développement. Afin d'introduire ces nouveaux concepts en Amérique du Nord, une municipalité du Québec va servir de modèle: la ville de Verdun. Dans ce chapitre d'introduction, nous préciserons d'abord quelques concepts de base et les éléments de la problématique. Par la suite, nous résumerons nos objectifs de recherche, les résultats attendus et nos principales contributions, avant d'esquisser les grandes lignes du mémoire.

1.1 Définitions et concepts de base

Afin de mieux saisir l'objet de ce mémoire, il convient de définir certaines notions de base nécessaires à la compréhension des infrastructures urbaines. Une infrastructure urbaine, aussi appelée *système urbain*, est un ouvrage ou une installation

qui offre des services de base à la population d'une ville. L'alimentation en eau potable, l'évacuation des eaux usées, l'alimentation en électricité en sont quelques exemples explicites. L'exploitation d'un système urbain se définit comme étant l'action qui consiste à mettre en valeur le patrimoine urbain, dans un souci de développement durable. Le développement durable fait référence à des modèles de développement qui répondent aux besoins des générations actuelles tout en préservant les possibilités de satisfaction des besoins des générations futures, du point de vue social, économique et écologique. Dans une optique aussi large, la gestion des infrastructures urbaines par les municipalités va bien au-delà de la seule prérogative des tâches d'entretien et de gestion quotidienne. En effet, il s'agit d'une réalité beaucoup plus complexe. La planification, la localisation, la réalisation, l'entretien des diverses composantes des systèmes urbains nécessitent la considération de plusieurs facteurs d'ordre géographique, démographique, technique, économique, « urbanistique », qui sont intimement liés à des préoccupations d'aménagement et de développement du territoire.

De cette idée est née la notion de gestion coordonnée de la ville par la mise à contribution des possibilités offertes par les technologies de l'information. Une telle approche est qualifiée d'*urbistique*. L'urbistique permet une gestion plus globale de la ville en tant que système, c'est donc une approche systémique[1] appliquée à la ville. Elle vise la maîtrise de l'information et de l'organisation à des fins de gestion intelligente ou rationnelle de flux qui parcourent la cité. Cette approche intégratrice entre les différents acteurs à l'échelle de la cité atteint son plein potentiel avec l'automatisation des processus et l'instrumentation des systèmes urbains. L'urbistique profite tout particulièrement des dernières avancées technologiques dans les domaines de l'automatisme, de l'informatique et des télécommunications.

[1] Méthode d'analyse et de synthèse prenant en considération l'appartenance à un ensemble et l'interdépendance d'un système avec les autres systèmes de cet ensemble.

1.2 Éléments de la problématique

De nos jours, en matière d'exploitation des infrastructures urbaines, il est fréquent que les décisions soient prises sans que tous les acteurs concernés ne soient consultés, sans que la réalité contextuelle qui précède une réalisation, de même que l'ensemble des conséquences qui s'en suivent ne soient clairement appréhendés. L'intégration de l'outil informatique n'est pas aussi fréquente dans les tâches quotidiennes. De plus, le concept d'intégration est très peu adopté par les concepteurs d'applications adaptées aux besoins de la gestion des infrastructures. Ces applications sont généralement de type propriétaire, ce qui rend coûteuses l'exportation des données et la mise à jour du produit. On constate l'existence de beaucoup de données brutes qui ne sont pas transférables, qui ne sont d'aucune utilité si elles ne sont pas transformées en informations de première importance permettant de prendre des décisions. Ces données sont réparties entre plusieurs applications qui, très souvent, n'ont aucune relation entre elles. Cela rend difficile la réalisation des concepts d'intégration dans un souci de développement durable.

Cependant, prenant appui sur les percées fulgurantes des nouvelles technologies de l'information et de la communication, un outil comme le SIGEC devrait répondre aux besoins présents et futurs des municipalités. Les bénéfices escomptés pour les utilisateurs sont très prometteurs, en voici un aperçu:

- augmentation de la présence et de la productivité du personnel sur le terrain ;
- diminution des efforts consacrés au support administratif ;
- meilleure affectation des ressources aux opérations sur le terrain ;
- augmentation de la proportion des efforts consacrés au préventif par rapport au curatif ;
- augmentation du ratio d'effort d'opération par rapport à l'effort de soutien ;
- visualisation systématique des interventions effectuées ;
- amélioration de la communication avec la clientèle, les partenaires et les ressources humaines ;

- diminution des coûts en ressources.

Dès lors, la disponibilité d'un outil puissant d'aide à la décision pour la gestion des infrastructures urbaines ne peut être que bénéfique entre les mains d'un gestionnaire de première ligne. Ainsi, celui-ci dispose d'une vision plus globale des actions touchant les opérations, la programmation, l'exécution et le suivi des travaux.

Par le principe même de système intégré de gestion coordonnée, le SIGEC représente un chef d'orchestre qui manipule l'information concernant les différentes infrastructures urbaines qui composent une municipalité. Il désire connaître et agir sur l'état actuel et futur (par projections et extrapolations des données actuelles) des systèmes urbains. C'est donc un outil d'aide à la décision et à la communication entre les différents acteurs qui interviennent au niveau des systèmes urbains dans leur ensemble. Outre la gestion coordonnée de plusieurs sous-systèmes urbains, la tâche qui consiste à exploiter un système urbain en particulier est laissée au SIDEX. Ce dernier est un outil plus spécialisé et mieux adapté au système urbain auquel il est assigné. Il s'occupe entre autres de la surveillance automatique à distance en faisant la saisie, la transmission et la transformation de données.

Pour concevoir ce système intégré qu'est le SIGEC, il faut d'abord définir le SIDEX générique qui permettra par la suite de l'appliquer, par instanciation, à différents systèmes urbains. Le grand défi qui se pose à nous est de concevoir les SIDEX de telle sorte qu'ils puissent favoriser une gestion dynamique ou en temps réel des infrastructures urbaines, en tenant compte du caractère évolutif de la répartition géographique de ces infrastructures. Il faut également disposer de réseaux de communications adéquats pour leur exploitation. La gestion dynamique que nous entendons ici constitue une approche de gestion et d'opération qui fait appel notamment à des équipements électroniques de contrôle de façon à optimiser la capacité et la fluidité du réseau et à en augmenter le niveau de sécurité. C'est donc une approche qui permet

de s'ajuster aux conditions réelles car porteuse d'une connaissance continue des systèmes urbains.

1.3 Objectifs de recherche

L'objectif principal de ce mémoire est de concevoir un système intégré d'exploitation (SIDEX) générique pour les infrastructures urbaines, répondant aux besoins des municipalités et capable d'être instantié pour n'importe quel système urbain. De manière plus spécifique, ce mémoire vise à :

- concevoir un SIDEX dédié à la gestion, à la surveillance et au suivi efficace des infrastructures d'égout de la Ville, SIDEX qui pourra être adapté à d'autres systèmes urbains ;
- définir la meilleure configuration possible pour installer sur certaines des infrastructures urbaines des automates et des équipements de télécommunications pour la saisie, le traitement et la transmission de données destinées à la prise de décision ;
- définir les mécanismes d'intégration de l'ensemble des SIDEX en un tout cohérent et convivial pour les différents types d'usagers du SIGEC, en fonction des tâches, des rôles et des responsabilités que ces usagers assument à la Ville.

Les aspects à prendre en compte lors du développement des différents systèmes sont les suivants :

- saisie des données ;
- transmission des données ;
- traitement et stockage des données ;
- conception des outils de simulation des systèmes urbains.

Ces travaux tiendront compte de tous les développements récents effectués dans le domaine de l'exploitation des infrastructures urbaines, dans un souci d'intégration, de normalisation de modèles de données et de développement durable.

1.4 Principales contributions escomptées

L'une des principales contributions escomptées reste la conception d'un SIDEX générique permettant aux municipalités de bénéficier d'un modèle conceptuel de données générique. Ceci constituera un apport incontestable dans la perspective d'éventuels travaux de normalisation dont il est de plus en plus question dans ce domaine. En effet, plutôt que de concevoir chaque SIDEX de manière indépendante, il est préférable de concevoir un modèle conceptuel de base qui tient compte de tous les éléments communs aux différents systèmes urbains.

Dans un autre ordre d'idée, le développement du SIDEX générique tiendra compte des aspects dynamiques de tous les systèmes urbains. La gestion en temps réel préconisée dans ce mémoire repose essentiellement sur les développements récents dans les domaines des télécommunications mobiles et fixes. Elle contribuera certainement à améliorer les processus de conservation et de réhabilitation des infrastructures urbaines.

1.5 Plan du mémoire

Ce mémoire comprend cinq chapitres. Le chapitre deux analyse les travaux récents déjà réalisés dans le domaine de la gestion intégrée des infrastructures urbaines. Le chapitre trois traite de la conception d'un prototype de SIDEX générique. Le chapitre quatre réalise l'instanciation du SIDEX générique pour le système urbain égout ou *assainissement* et traite de l'implémentation et de la mise en œuvre du SIDEX instancié. Finalement, le chapitre cinq, en guise de conclusion, fait une synthèse des travaux et esquisse quelques indications de recherches futures en vue de combler les inévitables lacunes de réalisation.

CHAPITRE II
INTÉGRATION DE COMPOSANTES RÉPARTIES

Le concept d'intégration est l'élément clé dans le développement des applications pour la gestion des infrastructures urbaines. Malheureusement, il est mal interprété et se traduit le plus souvent par des développements dispersés, répétitifs. Toutes les municipalités ne peuvent concevoir à elles seules toutes les composantes d'un système d'exploitation des infrastructures. Elles se basent sur des développements faits par d'autres entreprises et intègrent ces composantes (SIG, base de données, SIAD, système d'exploitation, etc.) au système de gestion courant qui n'est pas forcément informatisé. Cette étape reste tout de même laborieuse car ces municipalités ne possèdent pas les bases d'un SIDEX. Elle se traduit par des dédoublements d'informations et une incohérence du système vis-à-vis des tâches d'exploitation contrôlées par les gestionnaires. Ce chapitre synthétise les méthodes employées pour concevoir les systèmes d'exploitation d'infrastructures urbaines intégrant des composantes mobiles multimédias. Dans un premier temps, nous parlerons des besoins des municipalités et des systèmes existants dont elles disposent. Dans un deuxième temps, nous parlerons des technologies offertes qui sont présentement utilisés dans les applications mobiles multimédias et qui pourraient combler ce manque d'intégration au sein des municipalités.

2.1 L'exploitation selon les municipalités

Indépendamment d'une gestion coordonnée des infrastructures, les municipalités sont en recherche constante de données de toutes sortes provenant de diverses sources et permettant de déterminer et de prédire le comportement de leurs infrastructures à un instant donné. Certaines municipalités travaillent pendant plusieurs années pour

effectuer des mesures d'élevations sur le terrain, des inspections télévisées, des mesures de débit (Bégin, 1997). Malheureusement, ces données ne sont pas nécessairement exploitées par une seule municipalité. Des bureaux d'ingénieurs conseils de même que des organismes provinciaux ou fédéraux peuvent également produire ou utiliser ces données partagées.

Les municipalités ont plus tendance à avoir des systèmes décisionnels que des systèmes d'exploitation. Dans tous les cas, le premier a besoin du deuxième. L'aide à la décision doit faire intervenir la gestion coordonnée au niveau de toutes les infrastructures urbaines et non d'une seule ou de deux infrastructures. Pour bien exploiter une infrastructure, il faut minimalement connaître cette infrastructure. À un autre niveau, cette connaissance rend nécessaire de prendre en compte les caractéristiques du réseau et du tissu urbain, ainsi que la disponbilité des données (Blanpain et Karnib, 1998). Les municipalités ont besoin de connaître l'état structural (ou statique) de l'infrastructure à partir d'inspections vidéos, de relevés de conditions, de mesures sur le terrain. Elles ont également besoin de connaître l'état dynamique de l'infrastructure à partir des mesures de flux qui traversent les infrastructures (débit, charge). Toutes les municipalités ne possèdent pas un système automatique de mesure, ce qui peut être un handicap à la prise de décision éclairée. À partir des données de l'état statique et dynamique de leurs infrastructures, les gestionnaires des municipalités peuvent prendre des actions d'entretien, de réhabilitation, de planification et de conception. La Figure 2.1 présente un modèle typique de fonctionnement quant à l'exploitation d'une infrastructure urbaine.

Dans un système d'exploitation d'infrastructures urbaines, on retrouve entre autres le réseau d'aqueduc, le réseau d'égout ou d'assainissement et le réseau de la chaussée. Les bases de données et les SIG y sont d'une grande importance. La collecte et le traitement des données y sont souvent présentes mais restent tout de même disséminées (Côté, Lemieux, Fortier, 1997). Le marché offre des produits tout fait, adaptés à la gestion et à la maintenance des inventaires. Malheureusement, l'intégration de ces produits ne tient pas compte de la réalité des municipalités et des besoins des

gestionnaires, offrant ainsi peu de fonctionnalités adaptées à l'exploitation des infrastructures.

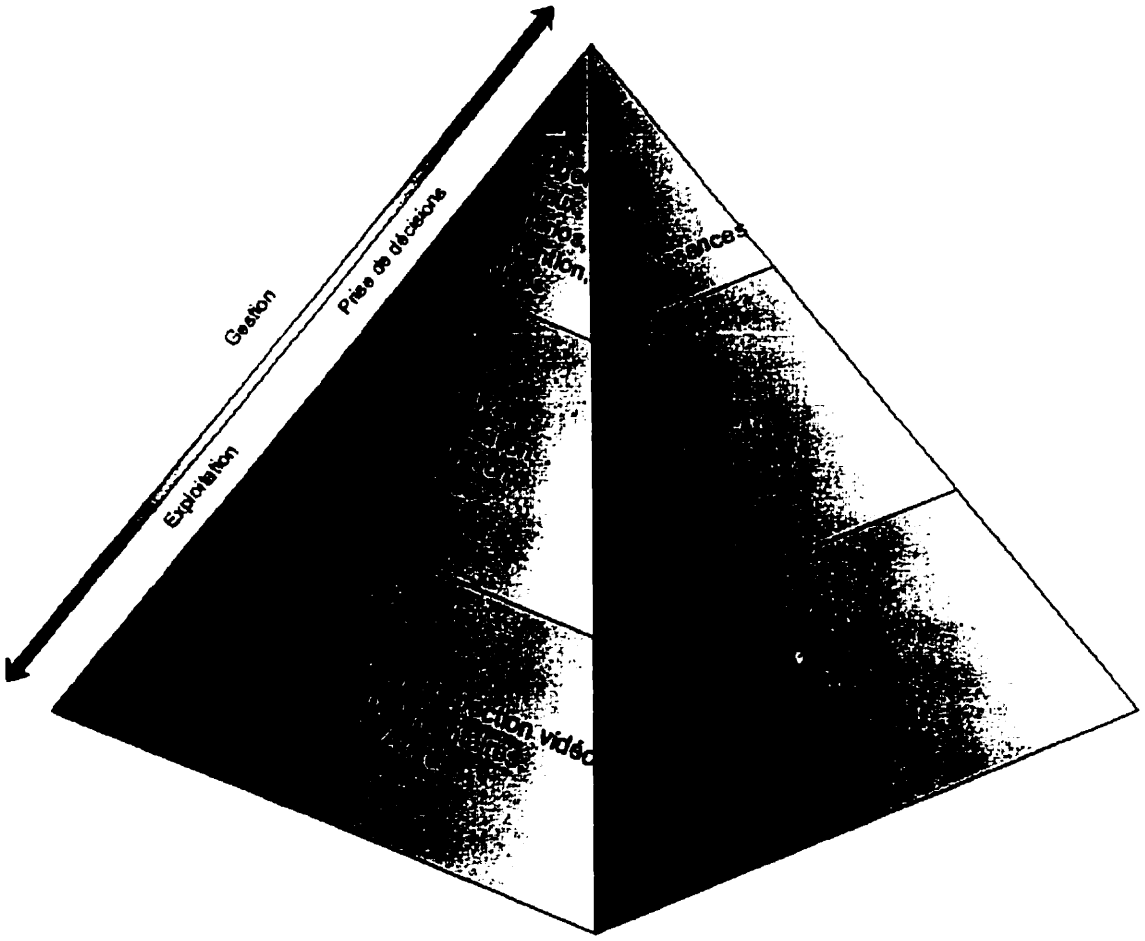


Figure 2.1 Fonctionnement des municipalités

L'exploitation d'une infrastructure consiste à transformer ces données brutes en information disponible d'une manière convenable aux gestionnaires.

> *« Data is raw information – a collection of facts that must be processed to be meaningful. Information is derived by associating facts within a given context », (Roger S., 1997).*

Par exemple, le nombre de plaintes concernant une infrastructure donnée est une information de première importance pour un gestionnaire. Cette information permet de prendre des décisions rapides (révision du programme d'entretien). Mais, lorsque cette information est jumelée avec d'autres, le gestionnaire peut alors prendre des décisions encore plus éclairée (programmation de budget par exemple). Toute cette chaîne allant de la collecte d'informations jusqu'à la prise de décision n'est pas une tâche facile, d'autant plus que les municipalités ne disposent pas d'outils adéquats. Ces derniers couvrent des domaines aussi variés que l'informatique (base de données spatiales, SIAD, système de classification), l'automatique (automates, contrôle) et les télécommunications (transission de données par réseaux). La tâche d'exploitation reste délicate à cause de la quantité abondante de données qui ne sont pas nécessairement compatibles (entendons par là, du même format). De plus, elles peuvent provenir d'organismes différents; donc la validité, et la précision des données ne sont pas assurées. Les municipalités ne possèdent pas de modèles tout fait pouvant les aider à normaliser. Cependant, des entreprises et des organismes proposent des solutions qui répondent plus ou moins aux besoins des municipalités. Dans la section suivante, nous présentons l'éventail restreint des solutions disponibles ou émergentes.

2.2 Outils d'exploitation

L'éventail des solutions proposées répondant aux besoins d'exploitation n'est pas très large. Le travail d'exploitation d'une infrastructure repose en grande partie sur les épaules du gestionnaire de première ligne. On retrouve des logiciels dédiés qui ne répondent qu'à une partie des problèmes des gestionnaires, des cadres de référence ou bases de travail qui sont des développements fait par des entreprises, organismes, centres

de recherche et utilisés par les municipalités pour développer des systèmes de gestion, d'information, d'exploitation et d'aide à la décision pour infrastructures urbaines. Tout compte fait, la construction de systèmes d'information robustes et évolutifs nécessite une réflexion approfondie et doit reposer sur des bases conceptuelles solides (Mottier, 1999).

2.2.1 Les logiciels dédiés

Pour répondre à leurs besoins de gestion et d'exploitation des infrastructures urbaines, les municipalités font appel à des solutions logicielles toutes faites. Parmi elles, on retrouve des logiciels de gestion des inventaires, des tâches, du personnel, des finances. Cependant, on constate trop souvent un manque d'intégration de ces logiciels désuets avec les outils déjà présents au sein des municipalités. Très souvent, ces logiciels ont une structure de données propriétaire à laquelle on ne peut apporter aucune modification. Par dessus cette structure, viennent se greffer des applications logicielles spécifiques (gestion de l'inventaire, des employés, etc). Ils n'intègrent pas de SIG, ce qui est un élément important dans le processus d'exploitation des infrastructures et de prise de décision par les gestionnaires des municipalités. Si les concepteurs n'ont pas prévu la gestion des plaintes par exemple, il sera presqu'impossible de réaliser ces tâches avec l'outil en question. Il faut alors acheter un autre logiciel uniquement pour la gestion des plaintes en se pliant aux exigences de ce nouveau logiciel. L'intégration de ce nouveau logiciel pour la gestion des plaintes avec les outils déjà présents pour la gestion de l'inventaire et des employés devient alors complexe. Il faut procéder aux étapes d'importation et d'exportation des données entre des logiciels possiblement concurrent et développer des applications prenant en compte la structure des données. Un accord sur les données d'entrées et la structure de ces données au début du processus de gestion des infrastructures facilitera alors l'échange de données entre les municipalités, les organismes et les entreprises partenaires. Les données spatiales ne sont généralement pas prises en compte car cela requiert la disponibilité d'un SIG. Bref, les municipalités qui utilisent ces outils tout faits en sont dépendantes. Lorsque le logiciel

évolue, elles devront elles aussi évoluer. Selon Mottier (1999), prendre un logiciel existant et l'adapter à des besoins spécifiques, présente un certain nombre de désavantages : de telles applications ne peuvent souvent pas être exactement adaptées aux besoins des utilisateurs.

2.2.2 Cadres de référence

Les cadres de référence sont des développements partiels faits par d'autres municipalités, organismes, entreprises ou centres de recherche afin de modéliser du point de vue informatique le processus de gestion des infrastructures urbaines. Ils sont souvent accompagnés de prototypes qui appliquent les concepts abordés. Cette nouvelle approche permet une standardisation des éléments intervenant dans la mise en place d'un système d'exploitation, de prise de décision ou de gestion en général dans le cas des infrastructures urbaines.

Dans le contexte de la gestion des infrastructures urbaines, un cadre de référence (framework) définit également une structure de données (à travers les objets du domaine d'application) qui supportera des applications dédiés à la gestion des infrastructures urbaines par les municipalités. Cela peut être un modèle conceptuel d'analyse, un modèle de conception ou même un modèle d'implémentation. Le modèle conceptuel d'analyse définit la gestion des infrastructures urbaines en général du point de vue logiciel. Il permet de définir les concepts du monde réel en des schémas, des pictogrammes et des mots, vu du point de vue logiciel et compréhensible par tous. Ces schémas peuvent montrer différents points de vue du processus de développement d'un logiciel dédié à la gestion des infrastructures en général. Ils peuvent montrer un modèle conceptuel d'analyse (le monde réel), un modèle objet (les concepts fondamentaux du domaine d'application visé par le système), un modèle d'implémentation (les concepts fondamentaux du point de vue de la programmation).

Les cadres de référence actuels sont conçus de manière ad hoc. Ils sont spécifiques à certains types d'infrastructure. Citons le cadre de référence développé par le CERIU (1999), qui est spécifique aux infrastractures d'aqueduc, d'assainissement et

de chaussée. Le cadre de référence ArcFM de l'ESRI (2001) est spécifique à l'assainissement et à l'aqueduc, tandis que le cadre de référence du VSA (1999) est spécifique à l'assainissement. Bien que certains d'entre eux soient conçus avec un langage de modélisation orienté objet (UML), l'inconvénient avec ces développements demeure leur manque de réutilisabilité et de généricité. En ce sens, bien que toutes les infrastructures urbaines aient un certain degré de similarité, ces cadres de référence sont trop axés sur un seul domaine à l'étude. En effet, lorsqu'on veut concevoir un système d'exploitation, un système d'aide à la décision ou un système d'information (particulier) pour l'éclairage, selon ces approches, il faut recommencer toute la conception à partir de rien. La plupart des cadres de référence utilisent le langage de modélisation UML sauf celui du CERIU. En effet, ce dernier utilise le concept entité-relation pour définir une structure de données normalisée. L'inconvénient majeur avec cela demeure l'impossibilité de représenter les traitements associés aux données, puisqu'un tel cadre ne définit pas la manière dont on doit manipuler les données pour offrir des fonctionnalités aux gestionnaires.

2.3 Architecture informatique

Les différentes méthodes d'implémentation sont intimement liées à l'architecture informatique. Il en existe plusieurs qui sont adaptées aux systèmes d'information et de gestion. Parmi elles, l'architecture répartie qui consiste en plusieurs noyaux s'oppose aux architectures centralisées possédant un noyau central fort. Les SIDEX, de par leur nature de système intégré, ont une facilité d'implémentation en systèmes répartis client/serveur. L'implémentation d'un tel environnement requiert l'intégration et la manipulation des concepts de base de données, d'interfaces graphiques, de systèmes transactionnels, d'objets répartis et également de mécanismes de communication. Afin de mieux saisir les étapes suivantes, il convient de préciser quelques termes importants et essentiels à la compréhension de l'architecture informatique proposée et des méthodes d'implémentation existantes.

2.3.1 Concepts de base

Le concept client/serveur est au cœur de toute architecture ouverte, il est d'autant plus flexible qu'il laisse le choix d'intégrer des composants hétérogènes. Il se compose de deux entités logiques qui accomplissent des tâches conjointement, à travers un réseau de communications. Entre les deux se trouve le « middleware » , qui est le lien entre le client et le serveur. C'est le substrat pour la plupart des applications clients/serveurs. Il commence avec les API qui se trouvent sur une station cliente et qui sont utilisés pour invoquer un service. Le « middleware » n'inclut pas le logiciel qui fournit le service (ceci se trouve chez le serveur), ni l'interface usager, et encore moins la logique de l'application. Il inclut les mécanismes de communication ou piles de protocoles, les RPC, etc. Il existe aussi des « middleware » plus spécifiques à certains environnements. On y trouve par exemple ODBC et JDBC pour la communication avec les bases de données.

L'architecture client/serveur est caractérisée, non seulement par les services fournis par les entités, mais aussi par des couches représentant la distribution de l'application entre le client et le serveur. La Figure 2.2 en est un exemple. Ainsi, lorsque la logique de l'application est répartie sur le client, on parle alors de « fat client », « thick client » ou encore « thin server ». Ce genre d'architecture est généralement utilisé lorsqu'on se trouve à l'intérieur d'un intranet ou lorsqu'on a un contrôle sur le client. Lorsque la logique de l'application est répartie sur le serveur, on parle alors de « thin client » ou encore « fat server ».

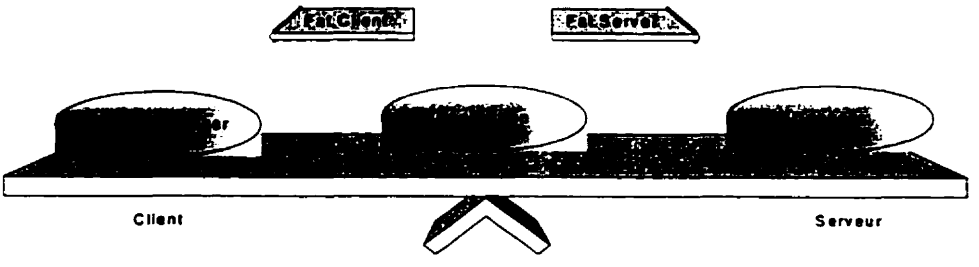


Figure 2.2 Exemple d'architecture client/serveur

D'autres termes peuvent être utilisés pour définir les mêmes concepts. L'idée principale reste la même, l'application est divisée en unités fonctionnelles que l'on peut assigner au client ou à un ou plusieurs serveurs. Les unités fonctionnelles les plus typiques sont : l'interface usager, la logique d'application, et les données. Il peut y avoir plusieurs autres niveaux dépendamment du découpage et du « middleware ». Dans les architectures 2-niveaux (2-tier), la logique de l'application est incluse soit dans l'interface usager chez le client, soit dans la base de données sur le serveur, ou encore sur les deux. Dans les architectures 3-niveaux (3-tier), la logique de l'application réside dans le niveau intermédiaire (middle-tier), séparée de l'interface usager et de la base de données. Ce dernier type d'architecture est considéré comme très flexible, robuste et évolutif.

La répartition de la logique de l'application en ces différentes couches a une incidence directe sur les performances de l'application, la sécurité ainsi que la réalisation de certains scénarios. Chacun de ces modèles a ses avantages et ses inconvénients, selon l'application développée. Dans bien des cas, les modèles se complètent et il n'est pas rare qu'ils coexistent dans une même application. Généralement, les architectures 2-niveaux se prêtent bien à des approches de prototypage RAD où la partie « visualisation » et la partie logique d'application se trouvent sur la même machine. Malheureusement, ce type de système ne présente pas d'API pour offrir l'accès à d'autres systèmes. Dans ce genre de système, en cas de changement de version, il faut par la même occasion faire une mise à jour chez tous les clients. De plus, comme la logique de l'application se trouve chez le client, la sécurité des données ne sera pas assurée. Quant à l'architecture 3-niveaux, elle supporte un meilleur découpage avec moins de surcharge du réseau.

2.3.2 Environnement informatique des municipalités

L'architecture informatique ainsi que les différentes méthodes d'implémentation, en plus de tenir compte des technologies actuelles, doit également s'attarder sur

l'environnement informatique des municipalités. Il ne s'agit pas d'élaborer un modèle parfait théoriquement mais pratiquement infaisable; il faut surtout tenir compte de la réalité des municipalités.

Un des buts visés est de faciliter l'accès aux données d'une infrastructure urbaine à partir d'une structure standard pouvant alimenter des systèmes d'aide à la décision pour le gestionnaire d'une municipalité. Un outil comme les SIDEX doit pouvoir s'intégrer à l'environnement informatique très hétérogène des municipalités. En effet, ces dernières ont investi énormément d'argent dans des outils informatiques spécialisés pour une infrastructure donnée au cours des dernières décennies afin d'assurer une gestion quotidienne et malheureusement non intégrée de leurs infrastructures urbaines. Plusieurs de ces outils font des tâches similaires, se retrouvent dans des services différents et mettent en évidence un manque de coordination évident. Par exemple, « Aquadata » est un logiciel de simulation du réseau d'aqueduc fréquemment utilisé par les services techniques, de même que « PCSWWMM » pour l'égout. La tâche de gérer les inventaires de tous les systèmes urbains est laissée à un logiciel tandis que la tâche de gérer les opérations courantes est déléguée à un autre logiciel. De cet état de fait résulte un dédoublement dangereux d'informations, ce qui peut être source d'incohérence à terme. D'un autre côté, les municipalités doivent faire face à ce besoin croissant de partage de l'information à distance avec des partenaires ainsi que les citoyens. L'architecture actuelle de leur environnement informatique ne répond pas à ces requis. Cette manière de fonctionner n'est pas conforme aux principes de la systémique sur laquelle repose le SIGEC incluant les SIDEX. Ces derniers se veulent être des outils de communication et de partage d'information afin d'optimiser le processus de prise de décisions. D'un autre côté, on ne peut changer radicalement l'organisation interne d'une municipalité sous prétexte qu'ils auront en mains un outil performant et avant-gardiste pour des fins de gestion coordonnée et de communication. Les coûts de formation et d'adaptation en feraient un échec total.

C'est cette approche intégrée, basée sur la systémique, qui est alors appliquée : intégration des SIDEX à l'environnement des municipalités, intégration de plusieurs

architectures standards, intégration de plusieurs outils ouverts incluant les technologies de l'Internet. L'un des outils les plus utilisés demeure le World Wide Web (WWW) communément appelé le Web. C'est une technologie Client/Serveur utilisée à l'échelle mondiale pour accéder à une grande variété d'informations. Avec un logiciel appelé navigateur (aussi appelé browser ou fureteur), tel Microsoft Internet Explorer ou Netscape Navigator et une adresse IP, on peut consulter des textes, des graphiques, du son et beaucoup d'autres types d'information présents dans presque n'importe quelle machine dans le monde qui agit comme un serveur. Le grand avantage est qu'on peut naviguer entre les différents documents (consulter plusieurs documents sur différentes machines) en utilisant les hyperliens et on peut aussi publier des documents, c'est-à-dire les rendre accessibles aux autres machines. Lorsque les technologies de l'Internet sont intégrées au sein d'une entreprise, on parle alors d'un Intranet. Un Intranet se distingue généralement de l'Internet par l'homogénéité des utilisateurs et la garantie de qualité de service du réseau.

2.3.3 Le client

Les clients se décomposent en deux grandes catégories : les clients légers (thin web client) et les clients lourds (thick web client). Ce sont des interfaces qui, selon le modèle, s'occupent des aspects présentation, validation et même exécution de la logique de l'application. L'interface est à base de menus implémentés la plupart du temps sous forme d'hyperliens déclenchant des opérations exécutées côté client ou côté serveur.

Les clients légers proviennent de l'architecture client/serveur (thin web client). Cette architecture est utilisée dans le cas où l'on a peu de contrôle sur le client, dans le cas où les ressources du client sont limitées. Elle garantit une configuration minimale qui permet de faire essentiellement de la consultation. C'est cette dernière qui est la plus utilisée dans le cas du commerce électronique, car elle cible une plus grande clientèle en permettant à des types d'utilisateurs différents de pouvoir acheter leurs produits. Ces entreprises qui font du commerce B2C ne peuvent se permettre de filtrer les requêtes des clients en fonction de leur navigateur. Ainsi, dans le cas du SIDEX, on considère les

navigateurs pour les clients de l'Internet à travers un réseau PSTN ou sans fil car ces clients sont limités en bande passante et ne sont pas homogènes. Font également partie de cette catégorie les clients WAP à travers des mini-navigateurs qui eux sont sévèrement limités en bande passante et en ressources (mémoire et processeur). Les navigateurs comprennent le langage HTML 3.0 standardisé respectant le DOM défini par le W3C. Le DOM est un API pour la conception de pages HTML et XML. Il définit la structure logique d'un document et la manière dont le document est accédé et manipulé. Avec le DOM, les programmeurs peuvent construire des documents de manière dynamique par programmation, ils peuvent naviguer dans leur structure, ajouter, modifier ou effacer des éléments ou du contenu. Sauf exception (éléments définis dans les toutes premières versions de HTML), n'importe quel élément contenu dans une page HTML ou XML peut être accessible (HEGARET, 2000).

Bien qu'ayant des comportements différents sur différents systèmes d'exploitation, les navigateurs en sont indépendants lorsqu'ils respectent les standards. Ces navigateurs doivent également comprendre le langage de script ECMAScript. Ces scripts sont nécessaires côté client pour faire le préfiltrage des requêtes, la validation des entrées ainsi que la mise en page. Ainsi, les versions 3 et suivantes des deux plus importants navigateurs, c'est-à-dire Internet Explorer et Netscape, satisfont à ces contraintes; ils ne seront donc pas obligés d'adopter les « cookies » pour la gestion des sessions. Les mini-navigateurs, quant à eux, doivent se plier au formalisme adapté aux mobiles, soit le standard WML et WMLScript. Les pages HTML, HDML et WML peuvent être générés automatiquement par les moteurs de script côté serveur.

Les clients lourds proviennent de l'architecture client/serveur « thick web client ». Tel que mentionné précédemment, cette architecture est utilisée dans le cas où l'on a un certain contrôle sur le client. Cela est réalisable seulement au sein d'une entreprise, dans l'Intranet ou l'Extranet. Par exemple, on peut supposer que, dans le cadre de la politique d'uniformisation informatique de l'entreprise, les clients seront tous des navigateurs Internet Explorer ayant des plugiciels (plugs-ins) bien définis. Ces composantes réalisent une partie de la logique des SIDEX. Une partie de la logique de

l'application est alors partagée avec le client qui se retrouve alors avec une interface beaucoup plus sophistiquée, par exemple pour la visualisation tri-dimensionnelle des éléments de système urbain. Dans certains cas, les données se trouvent directement chez le client, et les contrôles font de la validation. Comme exemple de validation côté client, la date d'abandon d'une conduite ne peut être inférieure à sa date d'installation. Dans certains cas, le client peut être un « viewer » propriétaire, qui se retrouve seulement dans le département des opérations/surveillance, tel un « viewer » pour des caméras et des automates. Ainsi, en plus d'implémenter les outils de l'architecture « thin web client », on ajoute de la logique à travers des scripts clients plus personnalisés, des applets, des contrôles et des plugiciels dans les navigateurs. Des contrôles tels Shockwave ActiveX peuvent faire des animations et des simulations très intéressantes. Les clients peuvent également utiliser *Microsoft Agent Control* pour accéder aux commandes de voix et exécuter des actions sur le navigateur, pour assister la navigation. Les scripts du côté client peuvent être *JavaScript* ou *VbScript*. On peut y retrouver des documents XML. Les contrôles *ActiveX* ou les *Javabeans* ont certains contrôles sur les ressources du client.

2.3.4 Le niveau-intermédiaire

Cette entité logique se décompose en plusieurs couches pour des raisons de performance. Les composants du middle-tier dépendent des systèmes d'exploitation. Par exemple, si l'on choisit un serveur web tel *Internet Information Server*, il est optimisé pour fonctionner sur les machines Windows NT; *Netscape Enterprise Server* est optimisé pour fonctionner sur les machines SUN, tandis que *Oracle Application Server* est optimisé pour fonctionner sur UNIX.

Au niveau de la première couche intérieure, on retrouve des éléments de filtrage des requêtes des clients. Ils jouent un rôle en regard de la sécurité des ressources de l'entreprise comme un coupe-feu (firewall), ou des rôles de conversion de format de données comme une passerelle WAP (relie le monde Internet au réseau mobile d'un opérateur).

Au niveau de la deuxième couche, on retrouve le serveur web, qui joue le rôle de serveur de contenu HTML pour les navigateurs ou de serveur de contenu WML pour les mini-navigateurs. Le serveur web est le principal point d'accès pour tous les clients navigateur. Les pages demandées peuvent être statiques ou alors créées dynamiquement. Selon la requête, le serveur peut initier des processus côté serveur. Pour des requêtes de pages scripts (ASP, PHP, JSP, Cold Fusion), le serveur web va déléguer la tâche aux modules exécutables CGI, ISAPI ou NSAPI. Les scripts CGI sont de moins en moins utilisés car leur performance est moindre par rapport aux autres modules, en ce sens qu'il crée un nouveau processus pour chaque utilisateur. Ces pages ont accès à toutes les ressources du serveur, incluant la logique des SIDEX sous forme de composantes, les bases de données, les anciens systèmes. Dans tous les cas, le résultat est une page HTML ou WML.

Àu niveau de la troisième couche, les serveurs d'applications sont des outils logiciels qui, pour des raisons de performance, résident sur des machines différentes du serveur web lui-même. Chaque serveur d'application joue un rôle bien précis. Il existe plusieurs types de serveur d'application qui sont regroupés dans les quatre grandes catégories qui suivent:

- Système d'exploitation : dans ce cas-ci, le serveur d'application est intégré au système d'exploitation. Microsoft soutient que son serveur Windows NT est un serveur d'application[2]. C'est un ensemble d'outils qui permettent la conception d'un serveur d'application robuste, accessible par les navigateurs et d'autres types de clients. En effet, ce dernier supporte les serveurs web (Internet Information Server), les serveurs de transactions (Microsoft Transaction Server), un serveur de bases de données (Microsoft SQL Server), un serveur de pages dynamiques (Active Server Pages).
- Intégré : Certains vendeurs incluent le serveur d'application dans les bases de données ou dans les serveurs web. Par exemple, Oracle 7 se compose de la base

[2] « Compaq, CSC, ISVs and Key Customers Highlight Growing Momentum For Microsoft's Application Server Technologies » Feb 24 1999, http://www.microsoft.com/presspass/press/1999/feb99/serverpr.htm

de données *Oracle*, muni du langage de programmation PL/SQL pour gérer la logique d'application et du serveur web *Oracle Web Server*.

- Plugiciel : ces derniers son similaires aux serveurs d'applications intégrés mais, au lieu de faire partie d'un autre produit (comme une base de données ou un serveur web), ils sont vendus séparément comme des composantes qu'on peut installer sur un autre serveur d'application. Cela leur assure quand même une certaine interopérabilité avec d'autres produits. Par exemple, la composante *MapX* qui offre des fonctionnalités d'identification géographique est un plugiciel vendu séparément, que l'on peut installer sur un serveur web, et qui peut collaborer avec une base de données relationnelle. On retrouve également dans cette catégorie *Tomcat*, *Webdev*, etc.
- Isolé (standalone) : ces derniers laissent le choix au concepteur d'application de concevoir son propre système sans être dépendant d'un produit spécifique. Ce genre de solution est très utile lorsque certaines composantes de base existent déjà, telles la base de données et le serveur web.

Les trois premiers types d'architectures de serveurs d'applications sont assez restrictifs car ils se basent sur des outils trop propriétaires. Un système d'information qui se veut ouvert ne peut être esclave d'aucun système d'exploitation ou d'un logiciel utilisant des protocoles non standards. Les SIDEX se feront avec différents langages de programmation et différentes technologies sur une plate-forme quelconque et ayant des relations avec les outils nécessaires à la prise de décision éclairés, outils qui se trouvent sur des serveurs.

2.3.5 Les bases de données

Les bases de données ont été depuis longtemps la fondation de toutes les grandes applications de systèmes d'information et de gestion. On y retrouve les données de l'application et également les données de la municipalité. Ces bases de données peuvent

être gérées par un ou plusieurs SGBD. Cependant, il faut tenir compte que chaque SGBD est propre à un système d'exploitation en particulier. Par exemple, *PostgreSQL* ne fonctionne que sur *LINUX*, *SQLServer* n'offre ses services que dans l'environnement *Windows NT*. De plus, les bases de données, comme les modules des serveurs web qui répondent aux pages scripts demandés par les clients, ont différentes architectures qui peuvent influencer les performances de l'ensemble de l'application.

Le processus par client est sécuritaire pour les usagers en ce sens qu'à chaque client est assigné un espace mémoire pour opérer sur la base de données. De plus, chaque processus peut être assigné à un processeur différent, dans un environnement multiprocesseur. Son désavantage est qu'il consomme beaucoup de mémoire et de ressources CPU (comme dans le cas du CGI). Il peut être lent à cause des communications inter-processus, mais ces dernières peuvent être gérées avec les moniteurs de transactions.

L'architecture « multi-thread » apporte de meilleure performance en assignant les connexions utilisateurs, les applications ainsi que les bases de données au même espace mémoire. Son seul inconvénient est qu'une application utilisateur mal intentionnée peut arrêter le serveur de base de données. Par exemple, des programmes utilisateurs qui font de longues requêtes peuvent ralentir considérablement le serveur de base de données. Finalement, la synchronisation préemptive du serveur de base de données tend à être inférieure à celle du système d'exploitation.

Les architectures hybrides se composent de plusieurs composantes qui sont des « listener » clients pour trier les clients, des queues et des processus. C'est une combinaison des deux architectures précédentes.

L'architecture « multi-thread » semble être une bonne solution pour les serveurs d'application en ce sens que la gestion des clients est laissée au serveur web (front-end) qui, lui, à travers un serveur d'application, va gérer plusieurs connexions pour plusieurs groupes d'usagers, plutôt qu'une connexion par usager.

La plupart des SGBD comprennent le langage SQL 89; les récents SGBD incluent également les standards SQL 92 et SQL 3 qui gèrent les procédures, les

déclencheurs et les règles. Ce sont des SGBD orientés serveurs d'applications. Ils implémentent certaines de leurs fonctionnalités en offrant des procédures qui vont manipuler la logique de l'application. La manière dont ces procédures doivent être programmées n'est pas standard. Les concepteurs de bases de données d'aujourd'hui rajoutent beaucoup d'extensions non-standards.

Les procédures cataloguées (stored-procedures) sont des mécanismes de type RPC pour appeler des procédures sauvegardées dans la base de données. C'est une collection de code SQL mélangé avec de la logique procédurale. C'est un objet de la base de données qui est enregistré et dont l'accès est contrôlé par le système de sécurité de la base de données. Le résultat est une réduction du trafic sur le réseau et de meilleures performances en termes de temps de réponse. Malheureusement, ces procédures sauvegardées ne sont pas assez flexibles, car non-standards. Chaque fournisseur à sa propre implémentation des procédures sauvegardées, du passage (par valeur ou par adresse) de paramètres des procédures, du langage de programmation utilisé (Oracle utilise par exemple PL/SQL).

Les « triggers » sont des exemples de procédures sauvegardées qui sont automatiquement déclenchés par le serveur de bases de données sur des évènements qui concernent certains types de données. Une règle est un type de déclencheur utilisé pour effectuer de simples vérifications sur les données. Les règles sont appelées implicitement lorsque certains évènements se produisent. Les implémentations des déclencheurs et des règles ne sont absolument pas standards.

2.3.6 Mécanismes de communication

La suite de protocoles TCP/IP, base des « middleware », assurera la gestion des mécanismes de communication. Les serveurs d'applications se trouvent en général à la couche *application* de cette suite et font appel à des protocoles comme HTTP pour desservir par exemple un client possédant un navigateur web. Ils s'occuperont également de la mise en place et du contrôle de dialogue avec d'autres applications, de la synchronisation et de la reprise après interruption. Le serveur d'applications s'occupera

de la compression syntaxique entre les utilisateurs en utilisant les standards tels ASN.1, mais aussi de l'encryptage des données en utilisant le protocole SSL. Certaines fonctionnalités qui sont demandées par les gestionnaires ne se trouvent pas directement dans la liste des outils qui collaborent avec le SIDEX. Il va falloir utiliser les sockets et recourir aux services des protocoles tels TCP ou UDP.

Au dessus de TCP/IP et SSL s'établit le protocole HTTP pour le web. Dans sa description officielle, HTTP est décrit comme un protocole de la couche *application* du modèle OSI pour des systèmes d'information répartis, intereliés et multimédia. C'est un protocole sans état, c'est-à-dire qu'après avoir envoyé une réponse à un client, il ne sauvegarde aucune trace de la connexion. Ce mécanisme doit être implémenté au niveau du serveur d'application pour assurer une sécurité minimale. HTTP est utilisé entre un client web et un serveur web.

Dans le cas des usagers mobiles, TCP/IP est remplacé par la série de protocoles orientés WAP utilisant les services de réseau mobile (GSM en Europe). Contrairement à HTTP qui ne gère pas les sessions des utilisateurs, dans le cas de WAP cela est géré par le protocole WSP. Ce dernier offre deux services de session. Le premier est un service connecté qui opère au-dessus de la couche *transaction*. Le second est un service non-connecté, qui agit au-dessus du service paquet (WDP). WDP est implémenté au-dessus du service des paquets et est orienté transactions. SSL et TLS sont remplacés par WTLS. WDP est le protocole de transport utilisé dans les réseaux mobiles. Il permet au service supérieur de faire abstraction du réseau mobile utilisé.

Des « middleware » plus spécifiques sont employés entre les clients et les serveurs d'applications et entre serveurs d'applications lorsqu'on se situe à l'intérieur d'un intranet pour des raisons de performance, quand on a un contrôle sur le client. Ils profitent des inconvénients de HTTP en utilisant par des mécanismes de communication orientés connexion tels RMI, IIOP et DCOM. Ce sont tous des protocoles qui sont implémentés au-dessus de la couche *transport* du modèle TCP/IP. Le navigateur est utilisé comme interface usager pour abriter des objets métiers qui vont communiquer indépendamment du navigateur avec des objets situés dans les serveurs d'applications en

utilisant les protocoles mentionnés précédemment. L'avantage d'utiliser un navigateur est que ce dernier possède déjà des fonctionnalités pour télécharger automatiquement les objets nécessaires. Un nouvel ordinateur n'a besoin de rien d'autre qu'un navigateur pour commencer à utiliser les serveurs d'applications. Ces composantes côté client sont des contrôles ActiveX, ou des applets Java; côté serveur, ce sont des servlets ou des dll.

Des « middleware » plus spécifiques sont également employés entre les serveurs d'applications et les bases de données. Les fournisseurs de SGBD ont leur propre API pour accéder à leurs données. Dans un environnement très hétérogène de données comme celui des municipalités, il est primordial d'avoir une interface commune à toutes ces bases de données gérées par des SGBD de différents fournisseurs.

L'architecture globale des systèmes intégrés pour l'exploitation des infrastructures urbaines sera présentée au Chapitre III.

2.4 Développement à base de composantes

Grady Booch (Larsen, 2000) définit une composante comme :

« a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces ».

il continue en spécifiant:

« an architectural pattern that provides an extensible template for applications within a domain ».

Une composante est une manifestation d'un objet du monde réel : c'est du code. Elle présente une interface bien définie derrière laquelle se trouve un ensemble d'implémentation. La composante peut être écrite en n'importe quel langage. Il est parfaitement possible de créer une composante en C. Ce n'est pas une librairie, en ce sens que la librairie n'offre pas une interface.

Certains auteurs (Hopkins, 2000) définissent les composantes comme un ensemble d'artefacts logiciels formant un tout cohérent pouvant être développé indépendamment et délivré en tant qu'unité. Ces composantes logicielles peuvent se composer d'autres composantes et collaborer entre elles pour bâtir quelque chose de plus

grand. Une composante peut se situer à n'importe quelle étape du processus de développement d'un logiciel.

La version 1.3 de UML définit une composante comme :

« a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces. A component represents a physical piece of implementation of a system, including software code (source, binary or executable) or equivalents such as scripts or command files ».

Les « frameworks », ou cadres de travail fournissent les éléments, relations, intégrités structurales fondamentaux permettant de bâtir des applications. Chaque « framework » devrait définir son mode d'extension. Un « framework » est une conception réutilisable de tout ou partie d'un système représenté par un ensemble de classes abstraites avec leurs interactions. Une autre définition est qu'un « framework » est le squelette d'une application qui peut être utilisée et arrangée par un développeur d'applications. Ces deux définitions sont plus complémentaires que contradictoires dans la mesure où la première est orientée *conception* tandis que la deuxième est orientée *implémentation*. Cette dernière est plus proche du code et donc du framework de « composantes ».

Le développement à base de composantes promeut la réutilisabilité (réutilisation de composantes préexistantes pour créer un système plus complexe) et la maintenabilité (les changements dans un système large seront simplement locaux et n'auront aucune influence sur le reste des composantes). De plus, le concept de composante est déjà bien implanté dans l'industrie dans plusieurs domaines. Un marché des composantes est déjà en place et permet essentiellement la réutilisation de code pour des applications générales. Composantes pour la connexion et la manipulation des objets d'une base de données ainsi que composantes pour la gestion des interfaces utilisateurs à une base de données en sont des exemples généraux d'applications, pouvant être utilisés dans n'importe quelle domaine.

Un exemple d'intégration de composante est la composante IMP pour la gestion des courriels destinés aux étudiants et employés de l'École Polytechnique de Montréal.

Cette composante est une série de scripts écrits en *PHP* et qui offre une interface générique. C'est une implémentation qui facilite le développement, elle s'intègre très bien à l'ensemble du système de gestion des employés et membres de l'école. Cette composante, bien sûr, se compose d'autres composantes encore plus spécifiques.

Idéalement, un développeur d'application devrait passer le plus fort de son temps à intégrer des composantes. Cela signifie qu'une composante doit être bien détaillée au niveau du modèle de classe et des cas d'utilisations. Les concepteurs de composantes doivent définir dans quelle mesure la composante pourra être modifiée.

Une attention particulière est nécessaire lorsqu'on travaille dans l'environnement client/serveur. L'implémentation des composantes côté serveur est faite de la même manière, quelle que soit l'architecture abordée. Dans un environnement Internet, on est intéréssé à la construction de composante spécifique au web : les pages webs. Les pages webs peuvent être implémentées avec des fichiers scripts (ASP, JSP, PHP) ou alors avec du code compilé (CGI, JavaServlet, ISAPI, NSAPI). Les pages compilées exécutent le rôle de plusieurs pages serveurs. Par exemple, un dll ISAPI pourrait être utilisé pour générer plusieurs pages dynamiques. La conception d'une seule de ces composantes est beaucoup plus complexe que n'importe quelle page dynamique. Les pages scripts, quant à elles, créent une composante par page. Il est possible pour une simple page script serveur de générer plusieurs pages clients.

CHAPITRE III
CONCEPTION DU SIDEX GÉNÉRIQUE

Au sens littéral du terme, le SIDEX générique est un système intégré dédié à l'exploitation d'une infrastructure urbaine générique. Les infrastructures qui sont à l'étude ont toutes des points en commun qui méritent d'être regroupés en un système représentatif de chacune des infrastructures, d'où le qualificatif de générique. De nos jours, la conception de systèmes aussi complexes que ceux qui font l'objet de ce mémoire ne se fait que selon une démarche de modélisation rigoureuse. Ce chapitre traite de la conception du SIDEX générique. Dans un premier temps, nous allons définir les propriétés caractéristiques et intrinsèques des systèmes urbains. Puis, nous exposerons l'architecture générale du SIGEC. Viendront ensuite les étapes d'analyse et de modélisation. En dernier lieu, nous verrons les mécanismes de surveillance et de communication présents dans le SIDEX générique.

3.1 Propriétés caractéristiques des systèmes urbains

Les systèmes urbains ont des caractéristiques qui leur sont propres et qui permettent de les distinguer de tout autre système. Il faut prendre en compte non seulement les aspects physiques (dimensions), structurels (composition), économiques (coût), et environnementaux (relation avec son environnement), mais aussi ceux reliés à son exploitation (rendement d'utilisation, interaction avec les acteurs).

Un système urbain est exploité par une organisation ou une municipalité. Leur travail consiste à aménager, planifier et gérer ces réseaux urbains. D'autres entreprises les aident dans cette tâche, ce sont des partenaires. On observe généralement une division administrative selon des modes de fonctionnement en plusieurs départements. Ces derniers s'occupent des entretiens et opérations courantes, de la conception et de la

construction ainsi que de la réhabilitation. Les personnes qui travaillent dans ces différents départements sont considérées comme des utilisateurs de premier plan des SIDEX et du SIGEC. Ce sont elles qui ont besoin d'outils d'exploitation, d'intégration et de coordination. Le citoyen est également considéré comme un utilisateur de second plan. En effet, il intervient comme un client recevant des services de l'infrastructure urbaine. Ainsi, toute personne a le droit de déposer des recommandations ou des plaintes sur les infrastructures et les services offerts.

Un système urbain se compose de plusieurs éléments connectés ou non en réseau qui possèdent chacun une propriété physique, une propriété structurale, un état de fonctionnement. Par exemple, le système urbain *égout* est un réseau interconnecté de conduites et de nœuds, tandis que le système urbain *bâtiment* est un ensemble d'éléments indépendants les uns des autres représentant des bâtiments.

La localisation et l'identification géographique d'un élément du système urbain sont deux aspects très importants qui interviennent tout au long de l'exploitation. En effet, l'identification est la propriété qui caractérise un élément à l'aide de ses coordonnées géographiques. La localisation est la propriété qui situe un élément dans son environnement (information topologique), elle se fait grâce à l'identification géographique. Par exemple, une conduite principale se trouve à gauche d'une autre conduite et ce, par rapport à un repère. Le choix d'un repère est primordial. L'Hôtel de ville peut en être un, mais généralement dans les zones urbaines les tronçons de rue et les intersections sont utilisés afin de faciliter les travaux quotidiens sur les infrastructures urbaines. Il est plus facile pour une personne de se rendre à l'adresse civique d'une conduite plutôt que de s'y rendre à partir de ses longitudes et latitudes.

Afin d'effectuer de la surveillance à distance, plusieurs équipements sont rajoutés sur toute l'étendue géographique des infrastructures urbaines. Ces équipements comprennent des instruments de mesure, des équipements de contrôle à distance et des équipements de télécommunications. Ces dispositifs servent à acheminer les données à analyser vers les municipalités afin de faciliter l'exploitation des systèmes urbains.

3.2 Architecture générale du SIGEC

Le SIGEC joue le rôle de chef d'orchestre pour tous les SIDEX qui sont exploités par une municipalité. La Figure 3.1 en présente l'architecture générale. Il se compose essentiellement de bases de données, d'une base de décisions/recommandations, d'interfaces de communication, et de systèmes intégrés d'exploitation pour différentes infrastructures urbaines.

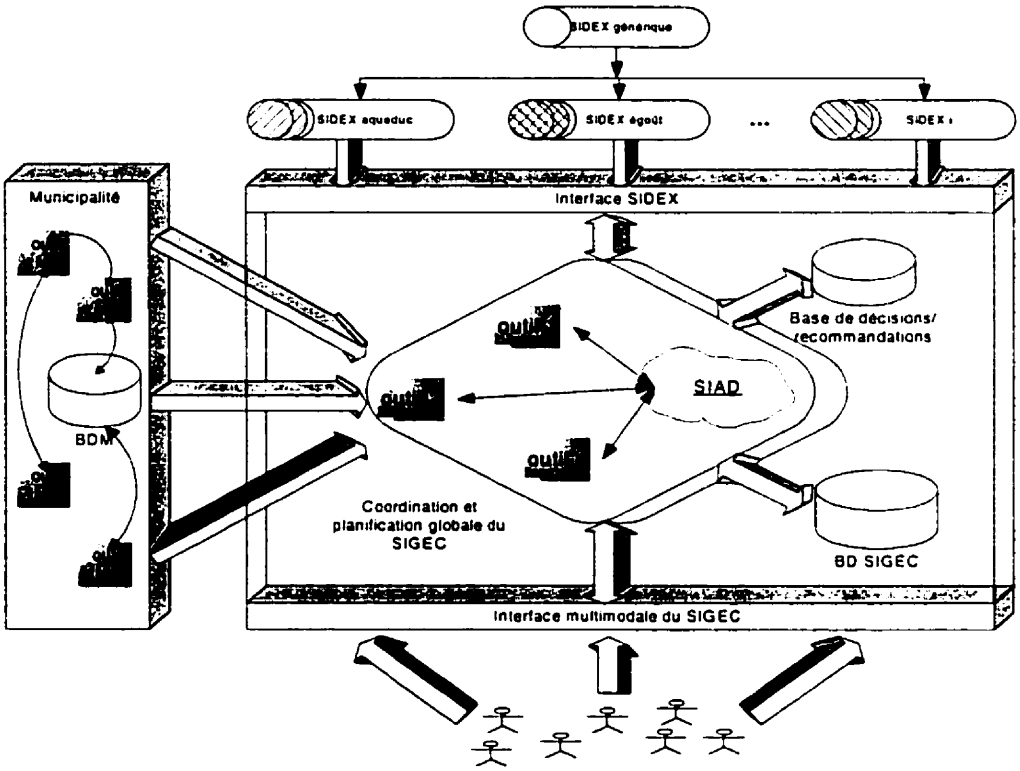


Figure 3.1 Architecture générale du SIGEC

Cette figure fait ressortir plusieurs sources d'informations. Certaines comme la base de données du SIGEC, et les bases de décisions/recommandations font partie du SIGEC; d'autres comme les bases de données de la municipalité (BDM) sont externes au SIGEC.

Généralement, les organisations possèdent toujours des systèmes reposant sur des bases de données qui sont à jour et qui répondent à la demande actuelle. Ces systèmes sont conçus pour accéder aux sources d'information de la municipalité selon une structure bien définie. Si des changements sont apportés à cette structure pour combler les besoins du SIGEC, cela risquerait de mettre gravement en danger ces systèmes anciens qu'on ne peut remplacer radicalement à cause d'un investissement de départ assez coûteux. Il faut donc créer une structure de données parallèle qui interagit avec les bases de données de la municipalité et qui alimente le SIGEC; d'où l'idée de base de données spécifique au SIGEC. Ces bases de données spécifiques au SIGEC sont utilisées lorsque des données sont partagées par plusieurs SIDEX.

Les outils ou composantes interagissent avec le SIAD à l'intérieur du module de coordination et de planification générale du SIGEC. Ce dernier est le cœur de l'application, en ce sens qu'il permet une gestion centralisée, coordonnée et intelligente de toutes les composantes qui l'entourent.

Les composantes utilisées ont des rôles bien précis; certaines possèdent une intelligence, d'autres non. Par exemple, il peut y avoir une composante pour la gestion des usagers du SIGEC. Son rôle serait de vérifier en tout temps l'identité de chaque personne qui désire effectuer une opération. Il devrait alors exister une opération *VérifierUsager* qui va consulter une base de données de sécurité et ainsi envoyer une réponse booléenne quant aux autorisations de cet usager.

Le rôle d'une autre composante serait d'adapter l'interface multimodale en fonction des caractéristiques du client (navigateur, paget, téléphone cellulaire, l'interface favorite de l'usager, ainsi que ses habitudes de travail). Il pourrait exister une autre composante qui, sous certaines conditions (déclencheur périodique, automatique ou manuel), va interroger chacun des SIDEX afin de récolter l'information sur les travaux futurs et va donner un ordre de priorité. Étant donné que le résultat de cette analyse concerne plusieurs SIDEX, il pourrait être sauvegardé dans la base de recommandations du SIGEC et être utilisé par tous les SIDEX. C'est lui qui manipule les composantes afin qu'il puisse répondre aux différents cas d'utilisation de l'application. Outre cette

intelligence de niveau élevé, le SIGEC se compose aussi des SIDEX. Chaque système urbain à ces caractéristiques particulières qui méritent d'être gérées par un système d'exploitation indépendant des autres. L'architecture des SIDEX sera définie dans la section suivante qui va traiter de son analyse et de sa conception.

Les SIDEX ont une dépendance les uns par rapport aux autres. En effet, dans la réalité, lorsque les gestionnaires ou cadres d'une municipalité exploitent une infrastructure urbaine, ils doivent tenir compte des infrastructures adjacentes. Ainsi, certaines fonctionnalités des SIDEX devront tenir compte des données et procédures se trouvant dans les autres SIDEX. Ces derniers peuvent s'échanger de l'information à travers leur interface, mais aucun ne peut altérer les informations de l'autre. Il s'agit de laisser en consultation les données propres au SIDEX et les procédures qui n'altèrent pas ces données. Les utilisateurs peuvent accéder à l'interface des SIDEX en passant d'abord par l'interface multimodale. Le but du SIGEC étant de créer une gestion coordonnée, la logique de l'application à travers une collaboration des composantes va s'occuper de la gestion des usagers et d'une sécurité centralisée au niveau du SIGEC, plutôt que d'avoir une sécurité répartie sur chaque SIDEX. Créer une base de données de sécurité propre à chaque SIDEX engendrerait beaucoup de redondance dans la mesure où plusieurs utilisateurs peuvent être amenés à travailler avec plusieurs systèmes urbains en même temps. Par la suite, l'interface configurable du ou des SIDEX concernés sera présentée à l'utilisateur.

3.3 Analyse et conception du SIDEX générique

Le processus de développement d'un système d'information est complexe. Afin de mieux maîtriser cette complexité, plusieurs méthodes de développement ont été proposées. Ces méthodes sont généralement organisées en un ensemble d'étapes représentant le cycle de vie du système d'information. Le cycle de vie propose un cadre conceptuel permettant d'organiser le processus de développement d'un système en le décomposant en sous-processus plus simples (Godin, 2000).

L'analyse et la conception sont des activités du cycle de vie dont l'objectif principal est de décrire la manière dont le système réalise les requis et les besoins spécifiés. L'analyse n'est pas un processus mécanique, mais plutôt un processus itératif que l'on retrouve dans le développement de tout système. Il n'existe pas de standard tout fait répondant aux systèmes ou aux phénomènes permettant de passer de manière formelle à travers ce processus. Par exemple, les étapes d'analyse pour la conception d'un véhicule diffèrent de celles d'une application bancaire. Cependant, des principes généraux existent et constituent un cadre général que l'on peut adopter au besoin. La phase de conception consiste en une optimisation, un raffinement de l'analyse. Dans le cadre de ce mémoire, les perspectives à considérer pour l'analyse et la conception du SIDEX générique sont d'élaborer les fonctionnalités détaillées en cas d'utilisation, ainsi qu'une vue logique qui contient les classes importantes et leur organisation en paquetages. Toutes ces étapes se complètent et sont nécessaires au processus de développement d'un SIDEX.

3.3.1 Cycle de développement

La conception des applications dédiées à la gestion des infrastructures urbaines suit, comme tout logiciel, un cycle de développement. Ce dernier se veut une méthodologie fiable pour le développement de produits répondant à un besoin bien précis. De nos jours, cette méthodologie est appliquée à un système logiciel pour une infrastructure donnée. La conception ad hoc de SIDEX entraîne une perte de temps et d'argent pour les municipalités ayant plusieurs infrastructures. En effet, les développeurs ne profitent pas de la réutilisation. La réutilisation est l'objectif ultime de toute approche appliquée au développement de logiciel et est basée sur l'idée qu'il est possible d'utiliser des artefacts logiciels existants dans de nouveaux contextes pour la création de nouveaux systèmes.

Cependant, une approche qui nous semble viable est la conception d'un SIDEX générique qui fait abstraction des différences pouvant subsister entre les différentes infrastructures. En effet, les systèmes urbains ont des caractéristiques communes qui

leur sont propres et qui permettent de les distinguer de tout autre système. Cette approche générique pour aborder la conception des SIDEX est à la base de la réutilisation.

L'approche orientée objet promeut la réutilisation, en ce sens qu'il offre des concepts plus adaptés que les méthodes fonctionnelles. En effet, l'abstraction, l'encapsulation, l'héritage sont des outils qui facilitent la réutilisation tout au long du processus de développement d'un logiciel. Ainsi, la conception du SIDEX générique suit une méthodologie orientée objet utilisant la technique OMT avec le langage de modélisation UML. Cette technique, adaptée aux besoins des infrastructures urbaines, propose un cycle de vie dans un cadre conceptuel permettant d'organiser le processus de développement d'un système en le décomposant en sous-processus plus simples. La Figure 3.2 présente les différentes étapes de conception du SIDEX générique.

À un premier niveau, le SIDEX générique représente une abstraction des SIDEX spécifiques à développer pour une municipalité. Son élaboration commence par une phase d'analyse qui est alimentée par les développements effectués sur les SIDEX existants. On entend par développement, toute étape du processus de développement d'un logiciel permettant d'effectuer tout ou partie d'un SIDEX (les patrons d'analyse, les « framework » de conception, les composantes). Dans le cadre des infrastructures urbaines, il existe de bons développements qui ont été effectués dans des domaines bien ciblés des infrastructures urbaines et qui ont servi à l'analyse du SIDEX générique. Tel que vu dans le chapitre précédent, les structures de données développées par le CERIU et le VSA ont servi à alimenter les différentes phases de développement du SIDEX générique, essentiellement l'analyse. Par la suite, un modèle de conception est développé. Ce modèle de conception est une évolution du modèle d'analyse, comme il se fait souvent dans certains processus de développement. Cela permet de maintenir au minimum le nombre de diagrammes, schémas et autres artefacts servant à la modélisation de l'exploitation d'une infrastructure urbaine.

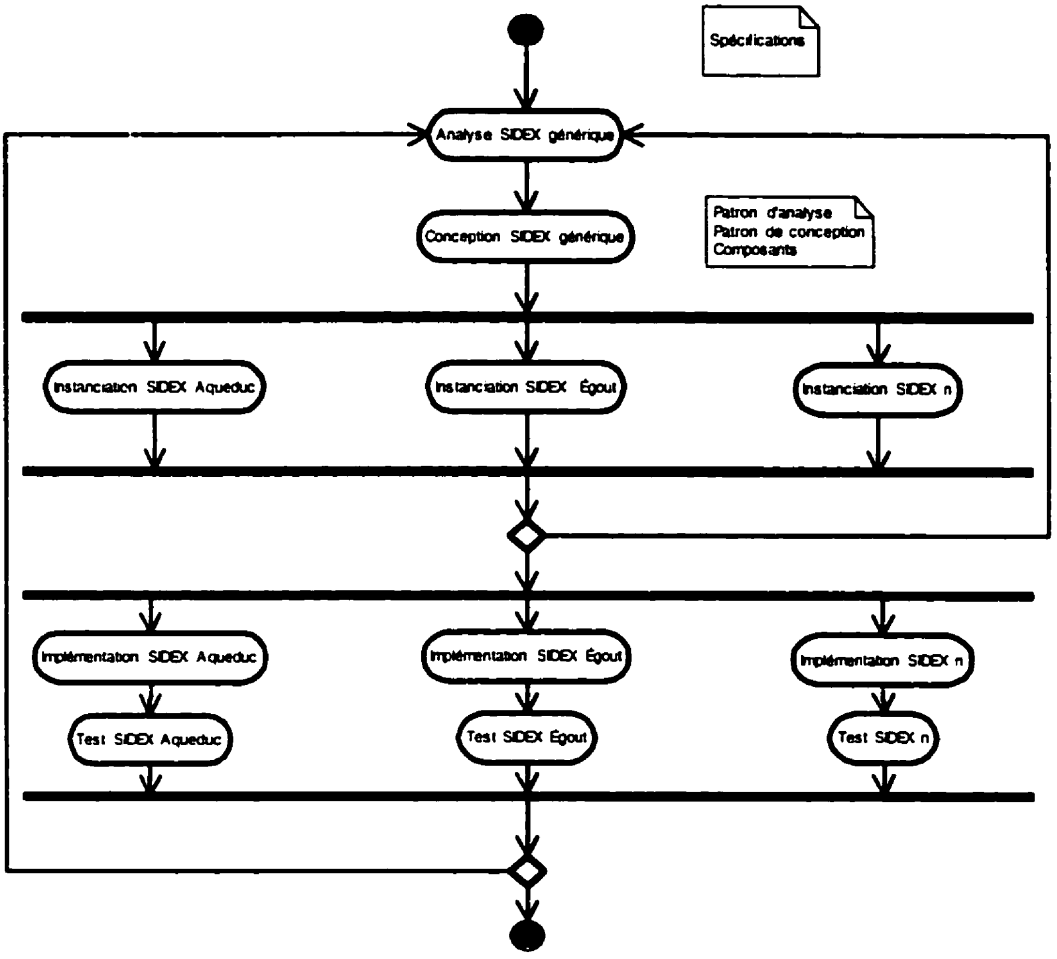


Figure 3.2 Cycle de développement du SIDEX générique

La prochaine étape est l'instanciation des différents SIDEX. Cette phase consiste à adapter le SIDEX générique à un système urbain spécifique. Cette étape nécessite bien sûr de connaître les éléments particuliers au SIDEX spécifique afin de pouvoir faire une analyse supplémentaire ainsi qu'une étape de conception. Le processus de développement des SIDEX spécifiques est alors moins long que celui du SIDEX générique. Ce gain de temps qui se manifeste par une facilité de développement basé sur un travail préliminaire est l'objectif recherché avec la généricité. La conception du

SIDEX générique n'est pas pour autant finie, elle se nourrit des développements subséquents de SIDEX spécifiques. Si après instanciation, on se rend compte qu'il y a certains éléments d'un système urbain qui n'ont pas été pris en compte, alors il faut déterminer s'il y a lieu de les rajouter dans le SIDEX générique afin que ces éléments soient profitables à la conception des autres SIDEX. La phase d'implémentation et de test est une étape normale de tout processus de développement de logiciel ou de prototype. Après cette étape et pour chaque SIDEX spécifique considéré, il faut revenir au SIDEX générique pour l'améliorer ou l'enrichir, selon une démarche itérative.

On remarque que le processus de développement du SIDEX générique est similaire au processus de développement de systèmes en spirale, à la différence que le SIDEX générique apprend des erreurs et bénéficie des développements des SIDEX spécifiques. Le développement de SIDEX repose donc sur un cycle d'apprentissage du SIDEX générique à chaque SIDEX spécifique et vice versa. C'est un élément très important de nos jours pour produire des applications dédiées à l'exploitation des infrastructures urbaines. Les municipalités ne peuvent se permettre de réinventer la roue, et de refaire les mêmes développements ou alors les mêmes erreurs. La connaissance de l'exploitation d'une infrastructure spécifique est un aspect important de réutilisation qui permet, en plus d'un développement rapide, une amélioration des SIDEX spécifiques.

3.3.1 Processus d'analyse

L'analyse du SIDEX générique a nécessité de s'imprégner de tous les concepts qui entourent les infrastructures urbaines. Il s'agit essentiellement de lire et relire les cahiers de charges, de se documenter et également d'avoir de longues discussions avec les spécialistes (gestionnaires, ingénieurs, superviseurs, etc.) des différents domaines qui entourent les systèmes urbains. Cela permet de se placer en tant qu'observateur privilégié sur chaque système et de faire une analyse la plus conforme possible à la réalité. Cette première phase de l'analyse fait ressortir les propriétés intrinsèques à chaque système urbain. Toutes les propriétés communes aux systèmes urbains sont regroupées en propriétés du système urbain générique qui servira de base au SIDEX

générique. Cette étape est décrite de manière textuelle en utilisant un langage rigoureux fait de phrases courtes et non ambiguës. Par la suite, afin de caractériser notre modèle idéal de SIDEX, nous avons défini les fonctionnalités de chaque système urbain.

Une fonctionnalité est une opération qui est réalisable par le SIDEX. Les fonctionnalités sont souvent très liées entre elles car elles peuvent en appeler d'autres dans le même SIDEX ou dans des SIDEX différents. Ainsi, les fonctionnalités communes à tous les SIDEX sont regroupées dans les fonctionnalités du SIDEX générique. En voici un échantillon :

- $O_{1.1}$ – Identification d'un tronçon de rue
- $O_{1.2}$ – Saisie d'information à références spatiales
- $O_{1.3}$ – Requête d'information à références spatiales
- $O_{1.4}$ – Tri par éléments fonctionnels et génération de tables relationnelles
- $O_{1.5}$ – Génération de la table de référence
- $O_{1.6}$ – Génération et accès à une banque d'images vidéo
- $O_{1.7}$ – Génération de formulaires automatiques complétés des télémesures
- $O_{1.8}$ – Validation des télémesures dans le réseau
- $O_{1.9}$ – Génération de la table des besoins et de la programmation des travaux
- $O_{1.10}$ – Interaction avec le système de gestion en temps réel
- $O_{1.11}$ – Génération des interventions et accès aux interventions prévus par des tiers sur des systèmes réseaux adjacents non directement gérés par la ville (NDGV)
- $O_{1.12}$ – Mise en correspondance de plans directeurs et de plans orthophotonumériques.

Une représentation plus complète de ces fonctionnalités figure à l'Annexe A.

Ces fonctionnalités définissent de manière globale ce que le système générique doit faire. Une des approches les plus répandues pour spécifier le système est la méthode

des cas d'utilisation (use cases) développée initialement par Jacobson (1992) et introduit dans UML lors de l'unification des langages de modélisation (1995). Ainsi, une fonctionnalité du système représente un cas d'utilisation, perçu par un utilisateur futur.

Un cas d'utilisation représente une façon particulière d'utiliser le système et correspond à une séquence de transactions reliées entre elles et produites par un dialogue avec l'acteur. Un cas d'utilisation décrit l'interface au système d'un point de vue de son utilisation par les acteurs qui sont les entités externes qui interagissent avec le système. L'identification des acteurs découle de la réponse à la question : « à qui est destiné le SIDEX? ». Les principaux acteurs sont des personnes qui peuvent utiliser le SIDEX à travers une interface (par exemple, la municipalité, les partenaires et les citoyens). D'autres acteurs interviennent dans le système tels les systèmes de gestion de bases de données, les outils d'analyses, les systèmes de gestion en temps réel à travers les réseaux de communication. Dans le SIDEX générique, les acteurs suivants ont été identifiés :

- *Usagers Internes* : utilisateurs au sein de la municipalité, lesquels sont divisés en catégories d'utilisateurs (administration, conception, construction, entretien et opérations). Ces utilisateurs auront accès au SIGEC afin de faciliter leurs tâches quotidiennes. D'autres utilisateurs externes auront simplement accès pour consultation ou surveillance ;
- *Usagers Externes* : utilisateurs qui travaillent dans des entreprises partenaires de la municipalité. Ils vont utiliser le SIDEX pour consultation des données ;
- *Clients* : citoyens utilisant les services fournis par la municipalité à travers les systèmes urbains. Ces derniers vont se servir du système pour s'informer ;
- *SGBD* : systèmes de gestion de bases de données du SIDEX qui s'occupent de la gestion des données internes et externes au SIDEX. Leurs fonctions sont de gérer les données dont le SIDEX a besoin ;
- *SIG* : systèmes d'information géographique qui offrent des services de représentation géographique au système de gestion de données référencées et également de requête à références spatiales ;

- *Systèmes de contrôle des équipements* : le SIDEX interagit avec des équipements de mesure et de contrôle qui prennent et acheminent les données jusqu'aux bases de données du SIDEX ;
- *Administrateur système* : l'administrateur du système s'occupe de l'installation et de la maintenance du système. Il doit entre autres créer les comptes pour les utilisateurs du système ;
- *Outils de simulation* : outils qui sont alimentés en données par le SIDEX pour effectuer des tâches spécialisées ;
- *SIAD* : système intelligent qui représente une source d'expertise.

Les cas d'utilisation sont accompagnés de documents complémentaires qui apportent une description claire. En effet, dans la phase d'analyse des besoins, toute ambiguïté doit être levée quant à la définition de certains éléments, aux flux d'évènements, aux exigences de performance, aux exigences de sécurité, aux exceptions. Le niveau de détail des cas d'utilisation peut varier, il n'existe pas de standard UML pour la description détaillée des cas d'utilisation. Cependant, il est utile de préciser les pré-conditions, les post-conditions, les conditions de déclenchement et de fin, la séquence normale d'interaction, les alternatives et les exceptions. Dans le contexte du développement de systèmes d'information typiques, une grande proportion des applications correspondent à des écrans interactifs (insertion, suppression ou modifications dans le cas d'une base de données) et des rapports (extraction de l'information utile). Chacun des écrans interactifs et des rapports peut être vu comme des cas d'utilisation. Dans le cas des SIDEX, ce ne sont pas seulement des systèmes d'information mais aussi un ensemble de systèmes d'exploitation des infrastructures urbaines qui n'ont pas souvent des écrans interactifs. Éventuellement, ces cas d'utilisation plus raffinés sont rajoutés et mis en relation avec les cas d'utilisation généraux. Les cas d'utilisation détaillés peuvent être définis de manière textuelle ou à

l'aide de diagrammes UML. Les deux ont le même rôle, sauf que l'utilisation de diagrammes est plus courante.

Les cas d'utilisation peuvent être reliés entre eux par des relations de généralisation et des stéréotypes <<inclut>> et <<étend>> :

- Relation de généralisation : permet de définir qu'un cas d'utilisation est une généralisation d'un ou de plusieurs cas plus spécifiques ;
- Relation de dépendance stéréotypée <<étend>> : permet de spécifier qu'un cas d'utilisation est considéré comme une extension du comportement d'un cas de base ;
- Relation de dépendance stéréotypée <<inclut>> : permet de spécifier qu'un cas d'utilisation utilise un autre cas d'utilisation.

Afin de définir de manière générique la documentation essentielle des cas d'utilisation du SIDEX générique, la structure suivante a été adoptée :

- Fonctionnalité : titre du cas d'utilisation ;
- Description : description courte de la fonctionnalité ;
- Acteurs : catégories d'utilisateurs représentant des entités physiques ou logiques ayant accès à cette fonctionnalité ;
- Flux d'évènements : aussi appelé processus, ces derniers sont utilisés pour représenter les cas d'utilisation avec tous les scénarios possibles. Les diagrammes d'activités d'UML peuvent servir à représenter des modèles de processus. Il s'agit de décrire ou d'utiliser les diagrammes d'activités d'UML afin de représenter les évènements déclencheurs des processus, les dépendances inter-processus. Toutes les règles particulières qui s'appliquent à ce flux d'évènements sont spécifiés. Toutes les alternatives doivent être spécifiées ;
- Définition des intrants et des extrants : ces derniers représentent les informations et matières consommées et générées par chaque processus

(les données manipulées par le cas d'utilisation). Un extrant d'un processus peut devenir un intrant d'un autre processus. Chaque intrant ou extrant se voit attribuer une valeur statique ou dynamique pour tenir compte de son évolution dans le temps. Un intrant statique signifie que les valeurs utilisées par le processus ne varient pas dans le temps. Un intrant dynamique spécifie que les valeurs sont mises à jour de manière manuelle, automatique et/ou périodique (venant des automates par exemple). Les valeurs possibles de l'intrant sont spécifiées ;

- Exigences de performance : s'il y a lieu, il faut spécifier dans quelle mesure une fonctionnalité pourra être réalisée, les ressources à utiliser, en combien de temps cette fonctionnalité doit être réalisée, afin de répondre aux critères de performance générale. Ces valeurs ne sont pas prises au hasard, mais doivent répondre aux exigences de l'utilisateur. Lors de l'implémentation, le nécessaire sera fait pour satisfaire ces exigences ;
- Exigences de sécurité : il s'agit de préciser les règles à adopter lorsque les acteurs désirent utiliser un cas d'utilisation. Par exemple, avant d'utiliser un cas d'utilisation, il faut présenter une identification unique de l'acteur ;
- Exceptions : ces derniers représentent les cas où les flux d'évènements sont interrompus.

La documentation détaillée des cas d'utilisation du module de gestion des plaintes du SIDEX générique figure à l'Annexe B.

3.3.2 Processus de conception

Le processus de conception du SIDEX générique aboutit à l'élaboration d'un modèle conceptuel. Ce dernier représente les concepts fondamentaux du domaine d'application visé par le système (appelé aussi modèle objet). Dans le cas des méthodes traditionnelles de développement, le modèle conceptuel pouvait se diviser en deux

parties : un modèle conceptuel de traitement et un modèle conceptuel de données. Avec des méthodes récentes de développement d'applications orientées objet comme UML, ces deux aspects sont intégrés dans le concept unificateur de classe. Cependant, dans les deux cas, il est préférable de commencer par l'identification des concepts du système d'information en se limitant aux données qui ont tendance à être plus stables que les traitements d'une application. Dans les applications du type système d'information, on se limite généralement à ne considérer que les données persistantes du domaine de l'application. Les données persistantes sont les données qui seront conservées à long terme et donc habituellement dans une base de données. Le long terme signifie entre autres que les données doivent survivre aux arrêts des programmes du système. À partir du modèle conceptuel du domaine, les concepts qui doivent être maintenus de manière persistante sont identifiés à l'aide d'une valeur étiquetée "persistant". Le développement du modèle conceptuel est un processus itératif de raffinement graduel qui implique une séquence d'essais/erreurs (Godin, 2000).

Pour commencer le processus de conception, la description des fonctionnalités du SIDEX générique constitue un bon point de départ. En effet, cette première étape d'analyse permet de faire ressortir les classes et instances de classe (les objets). Si l'on considère la fonctionnalité $O_{1.1}$, on constate qu'il y a plusieurs tronçons de rues dans le système urbain géré par le SIDEX générique. Chaque objet (par exemple un tronçon de rue) est unique; ainsi, il est donc préférable de créer la classe *TronconRue*. Chaque tronçon de rue sera donc une occurrence de la classe *TronconRue*. Ces objets seront générés par instanciation de la classe mère. Les classes correspondent souvent à des noms. Par exemple, lorsqu'on lit l'énoncé suivant qui découle des propriétés caractéristiques du SIDEX générique : « Les clients qui bénéficient des services offerts par la mise en place de ses infrastructures peuvent déposer des plaintes et des réclamations », clients, plaintes, réclamations sont des classes potentielles. Par la suite, on procède à la sélection des classes par élimination des classes non pertinentes.

L'étape suivante est l'identification des associations entre les classes. Les associations correspondent généralement à des verbes ou expressions verbales. L'énoncé

« Un système urbain est géré par une organisation ou une municipalité » qui découle des propriétés intrinsèques aux systèmes urbains (cf. section 3.1) permet d'identifier l'association qui lie un système urbain à une organisation. Les bonnes associations sont sélectionnées par élimination des associations entre classes supprimées, des associations non pertinentes ou d'implémentations, des actions, des associations ternaires, des associations dérivées, des associations mal nommées, des noms de rôles, des associations qualifiées, des multiplicités, des associations manquantes.

Viennent ensuite l'étape d'identification des attributs pertinents. Ces derniers sont propres à une classe donnée et sont sélectionnés par élimination des objets, des qualificatifs, des noms, des identificateurs, des attributs de liens, des valeurs internes, des détails fins, des attributs discordants.

Par la suite, le modèle est continuellement raffiné. Les concepts de *généralisation* et de *spécialisation* sont appliqués afin de regrouper les structures communes en une seule classe. La dernière étape de la modélisation consiste à regrouper les classes en « packages ». Ce découpage doit être guidé par les considérations de couplage et de cohésion. Ces « packages » ne sont pas complètement indépendants les uns des autres, ils ont des liens à travers les classes.

Afin de ne pas altérer la connotation des classes et d'en donner une interprétation multiple, un dictionnaire des données est créé au fur et à mesure de l'élaboration du diagramme de classes. Il est constitué d'une description textuelle des attributs, des opérations et des associations.

3.4 Architecture du SIDEX générique

Le SIDEX générique est une abstraction du monde réel. Son objectif est d'exploiter un système urbain fictif. Illustrée à la Figure 3.3, l'architecture du SIDEX générique est le résultat de l'analyse effectuée à l'étape précédente, c'est l'organisation complète du système. Elle intègre les « packages » *ORGANISATION*, *USAGER*, *INTERFACEDONNEES*, *INVENTAIRE*, *IDENTIFICATIONGEOGRAPHIQUE*, *Mesure* et *Travaux*. Avant de passer à la description détaillée de chacun de ces sous-systèmes, il est

important de définir la sémantique utilisée. Certains de ces packages se situent entre les SIDEX et le SIGEC (IDON, USR, ORG, IGEO, PLT, TRV). Ils ont des données qui sont partagées par tous les SIDEX et le traitement associé à ces données est identique d'un SIDEX à l'autre. Si les traitements associés aux données sont différents d'un SIDEX à l'autre, alors ils n'ont pas besoin de se situer au niveau du SIGEC mais bien du SIDEX.

3.4.1 Sémantique

Voici, la sémantique employée dans le modèle conceptuel :

package : Un package est une subdivision d'un système dans le langage UML. C'est un regroupement d'éléments ayant des comportements similaires dans un sous-système.

Classe: représente un ensemble d'objets avec une structure et un comportement similaire.

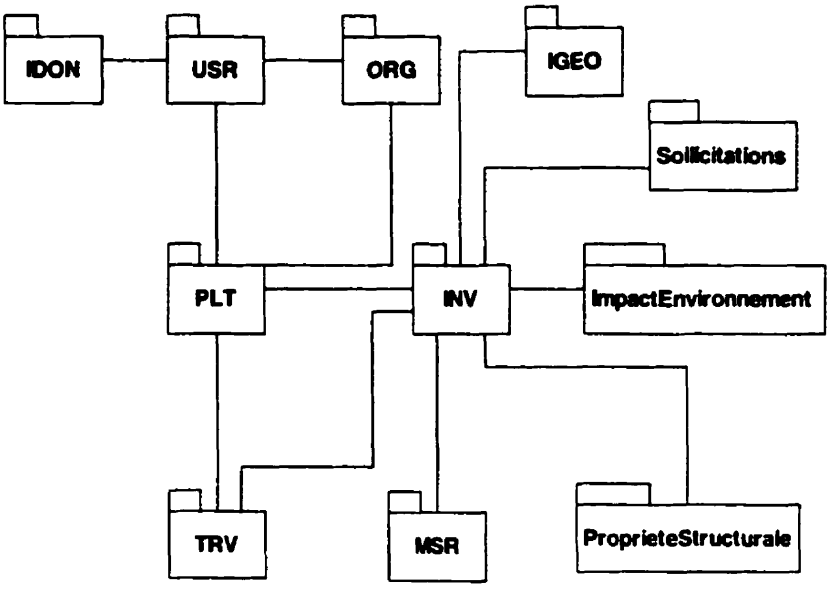
Association binaire: représente une relation entre deux classes.

Multiplicité: représente le nombre d'objets pouvant participer à une relation exprimée par un nombre ou un intervalle croissant.

Rôle: indique le rôle joué par une classe dans une association.

Généralisation/Spécialisation: représente une relation entre un élément spécifique et un élément général de telle manière que l'élément spécifique soit en accord avec l'élément général et contienne plus d'information.

Agrégation/Composition: Association asymétrique dans laquelle un objet d'une classe est considéré comme le tout et les objets de l'autre classe sont considérés comme les parties.



Légende:

IDON:INTERFACE_DONNEES
USR: USAGER
ORG: ORGANISATION
IGEO: IDENTIFICATION_GEOGRAPHIQUE

PLT:PLAINTE
INV: INVENTAIRE
TRV: TRAVAUX
MSR:MESURES

Figure 3.3 Architecture logique du SIDEX générique

3.4.2 Package ORGANISATION

Ce sous-système comprend les classes nécessaires pour représenter l'ensemble des organisations qui participent à la tâche d'exploitation des infrastructures urbaines par une municipalité. La Figure 3.4 montre l'organisation des classes persistantes de ce « package ». Un organisme, du point de vue générique, est caractérisé par les attributs minima suivants : *nom*, *adresse*, *telephone* et *Enregistrement*. C'est une classe abstraite qui se compose de plusieurs départements. Du point de vue d'une municipalité, cet organisme peut représenter la municipalité ainsi que les partenaires. Lors de l'instanciation, pour un SIDEX spécifique dédié à une municipalité, d'autres attributs permettront de mieux caractériser un organisme. Ce package se compose également de

différentes règles, normes et références concernant l'infrastructure urbaine. Ces classes servent à connaître les réglementations en vigueur sur chaque élément d'inventaire. La raison d'être des éléments de référence dans le SIDEX générique est que les références sont propres au SIDEX auquel ils sont rattachés. Il n'y a pas de références communes à toutes les infrastructures urbaines en même temps. Éventuellement, de nouvelles règles et références pourront se rajouter automatiquement, selon la logique qui guide le fonctionnement de la base de décisions ou de recommandations locale. Cet ensemble de règles sert à alimenter un SIAD.

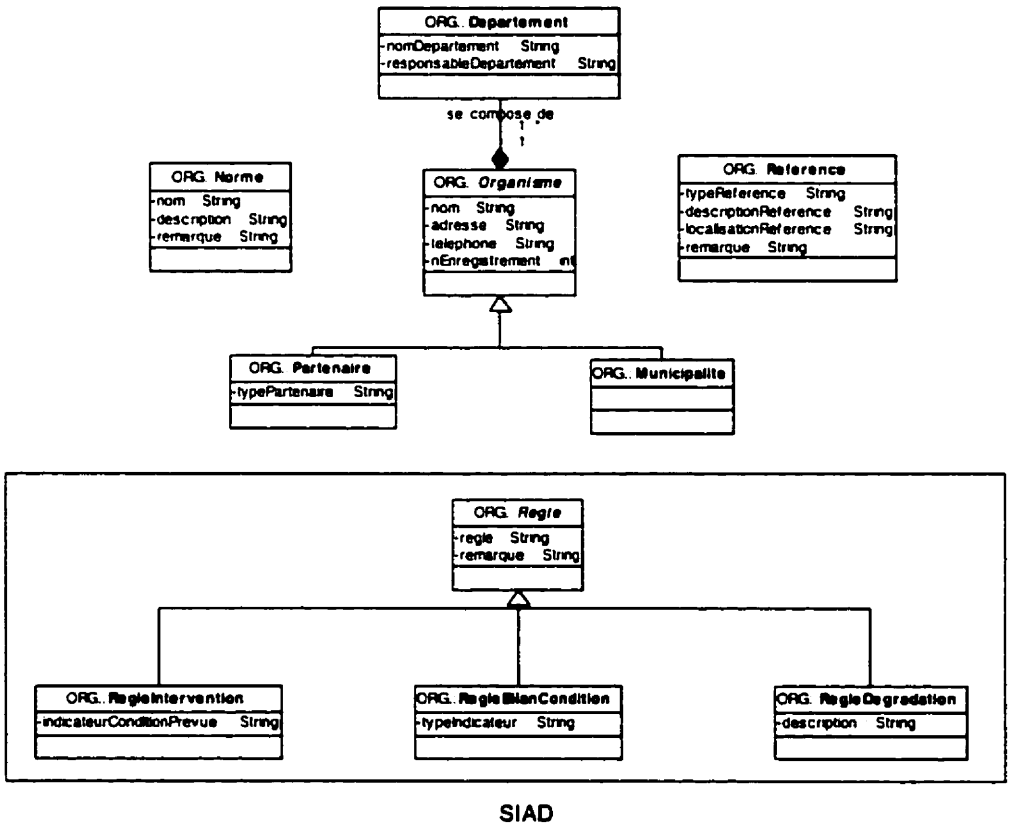


Figure 3.4 Package ORGANISATION

3.4.3 Package USAGER

Ce sous-système se compose des différentes classes regroupant l'ensemble des usagers qui interagissent avec une infrastructure donnée. La Figure 3.5 montre l'organisation des classes de ce « package ». On y retrouve plusieurs types d'usagers qui se retrouvent dans des groupes d'usagers. On distingue les employés des citoyens. Ces derniers font également partie des utilisateurs du système, ils ont le droit d'émettre des plaintes sur une ou plusieurs infrastructures urbaines. Tous ces types d'utilisateurs découlent de la classe abstraite *UtilisateurGénérique*. Un utilisateur possède un profil qui permet de gérer tout un ensemble de couples clé/valeur. Par exemple, un utilisateur qui gère les plaintes aimerait avoir une couleur de fond d'écran différente de celle définie par défaut. Lors de l'exécution du programme, il y a chargement des données persistantes au niveau de l'application pour traitement. S'il est implémenté en HTML, le programme va simplement lire la clé *bgcolor* (tag HTML pour fixer la couleur de fond) et va lui attribuer la valeur spécifiée par cette clé. Ainsi, cette classe permet de configurer l'interface d'un utilisateur selon ses besoins. Les utilisateurs peuvent faire partie de plusieurs groupes d'utilisateurs et tous les groupes sont administrés par le groupe des administrateurs du système : *AdministrateurSysteme*. Chaque SIDEX possède plusieurs types d'utilisateurs ayant un certain nombre d'opérations leur permettant d'interagir avec le système. Les groupes peuvent se composer d'autres groupes. Si plusieurs SIDEX sont instanciés, alors ce « package » se retrouve au niveau du SIGEC. Car, pour des raisons de sécurité, la centralisation des données et des traitements est préférable à un niveau global plutôt qu'à un niveau local. Ce dernier cas est possible, mais extrêmement complexe à réaliser à cause des dédoublements d'information entre chaque SIDEX.

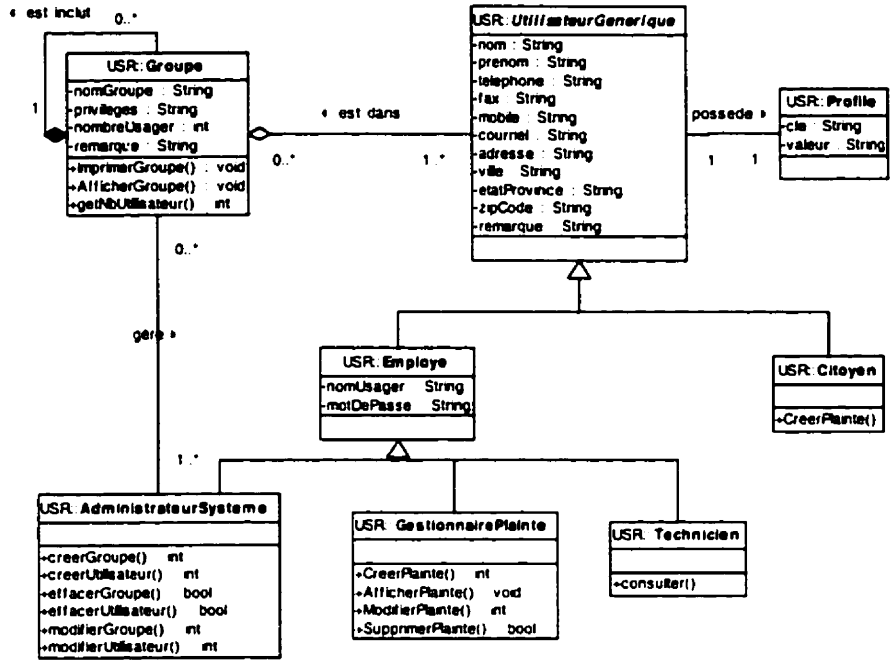


Figure 3.5 Package USAGER

3.4.4 Package INTERFACE_DONNEES

Ce sous-système intègre toutes les classes des interfaces qui se présentent à un utilisateur. Les utilisateurs du système n'ont pas directement accès aux ressources. Le package *INTERFACE_DONNEES* implémente l'interface qui sépare les utilisateurs des ressources. La Figure 3.6 montre l'organisation des classes de ce « package ». Lorsque les utilisateurs accèdent au système, une session est démarrée. Cette dernière est représentée par un identifiant unique, une heure de démarrage et également une heure de fin. Cette classe permet de configurer les variables d'environnement du système. La classe *Historique* implémente la notion de gestion et de surveillance des évènements des utilisateurs. Ainsi, tous les évènements importants qui sont générés lors de la session d'un utilisateur sont sauvegardés dans le système. Ce « package » contient également un ensemble de classes permettant l'accès aux données de n'importe quelle base de

données. Cette couche d'abstraction aux données est un élément très important dans l'accès aux données persistantes qui peuvent se situer sur n'importe quelle machine.

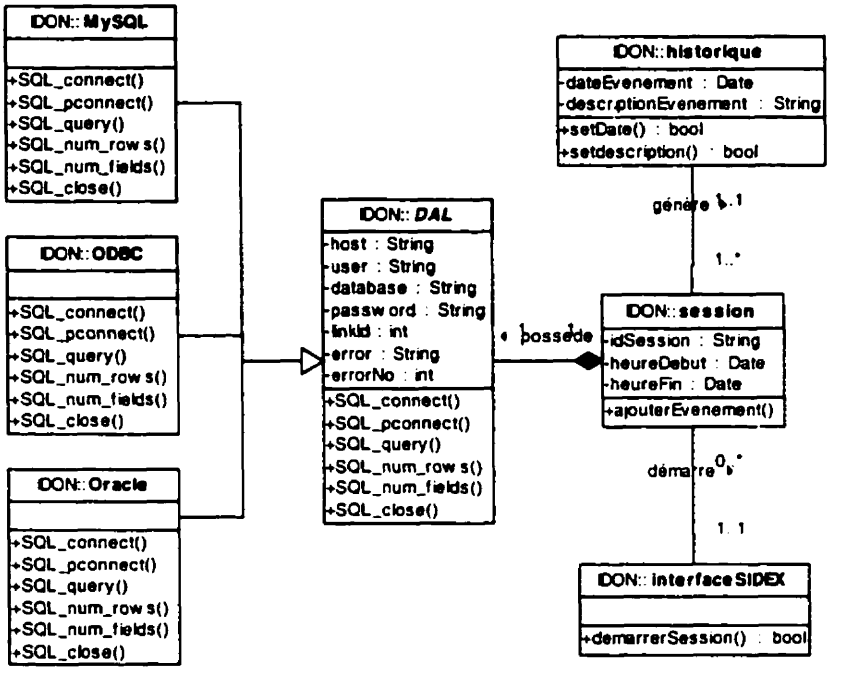


Figure 3.6 Package INTERFACE_DONNEES

3.4.5 Package PLAINTE

Ce sous-système gère l'information sur les requêtes en général. La Figure 3.7 montre l'organisation des classes de ce « package ». Une requête peut être une plainte, une recommandation ou une suggestion (pour augmenter la qualité de service par exemple) concernant une ou plusieurs infrastructures données. Si la requête est une plainte, elle peut engendrer des réclamations, définies par la classe *Reclamation,* ainsi qu'un traitement défini par la classe *Traitement.* Pour une plainte donnée, il est possible de trouver des plaintes similaires, basées sur les mêmes valeurs des attributs. Ces plaintes similaires, si elles ont eu un traitement particulier, permettront d'alimenter un SIAD et de prendre une décision concernant la plainte en question. Si le SIDEX

générique est instancié en plusieurs SIDEX pour une municipalité donnée, alors ce « package » doit se situer au niveau du SIGEC. Il faut concevoir qu'une plainte peut concerner plusieurs infrastructures de manière simultanée même si, en pratique, cela peut s'avérer très rare. La sauvegarde générale ainsi que le traitement des plaintes se situent au niveau du SIGEC.

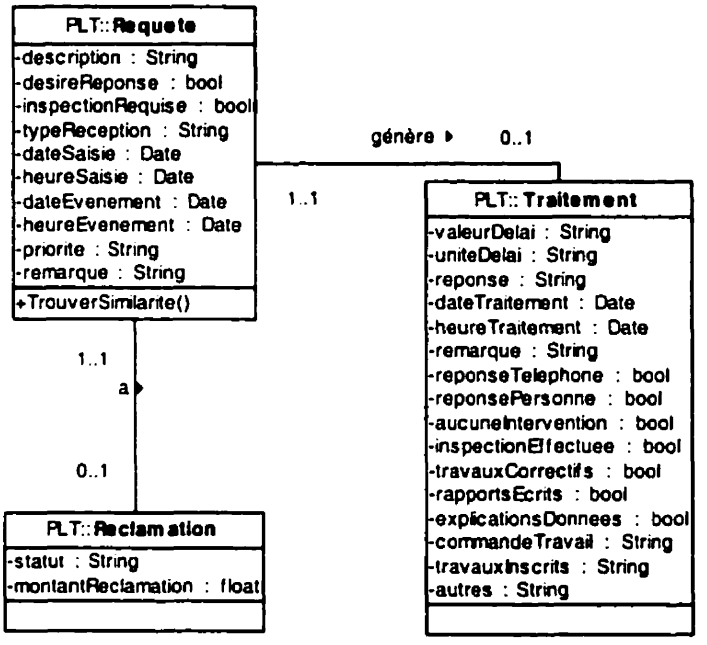


Figure 3.7 Package PLAINTE

3.4.6 Package IDENTIFICATION_GEOGRAPHIQUE

Tel que recommandé dans les fonctionnalités du SIDEX générique, ce sous-système contient l'information nécessaire pour pouvoir identifier géographiquement chaque élément d'inventaire du système urbain. La Figure 3.8 montre l'organisation des classes de ce « package ». Une section de route (*SectionRoute)* se décompose en tronçon de rue (*TronconRue*), intersection (*Intersection*) et section fictive (*SectionFictive*). Plusieurs sections de route forment une route. Cette dernière est prise dans le sens d'une

voie terrestre continue praticable par les voitures; elle est reliée à une autre route. La classe *Parcelle* permet d'identifier les portions de terrain bâti ou à bâtir qui se situent en bordure d'une route.

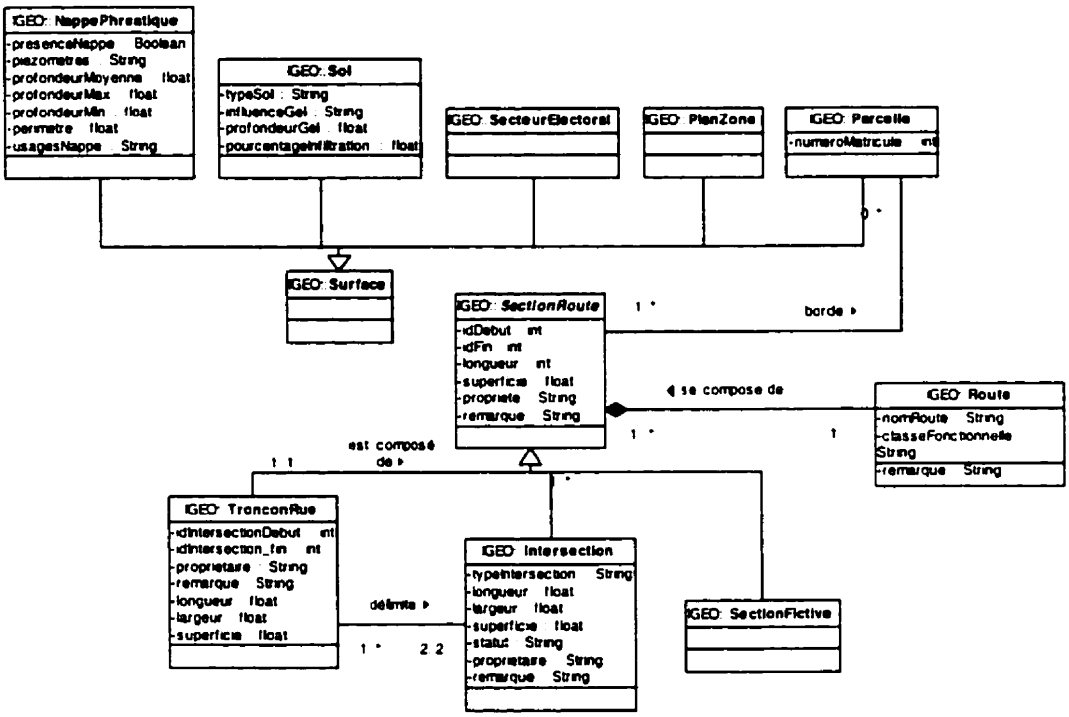


Figure 3.8 Package IDENTIFICATION_GEOGRAPHIQUE

Le concept de section est défini par les classes *SectionRoute* et *SectionFictive* pour raffiner les requêtes sur l'interrelation des tronçons de système urbain avec les tronçons de rue. Physiquement, les tronçons de rue représentent un rectangle, mais lorsque l'on veut représenter une fraction de tronçon ou un ensemble de tronçons, alors il faut définir la section. Cette dernière est une fraction d'un élément de la chaussée. La Figure 3.9 présente le concept de section dans la chaussée. L'avantage principal est que l'information utile est regroupée dans une section. Ce concept n'est pas obligatoire mais

simplifie beaucoup les coûts de réfection des chaussées dus à la réhabilitation des éléments de système urbain.

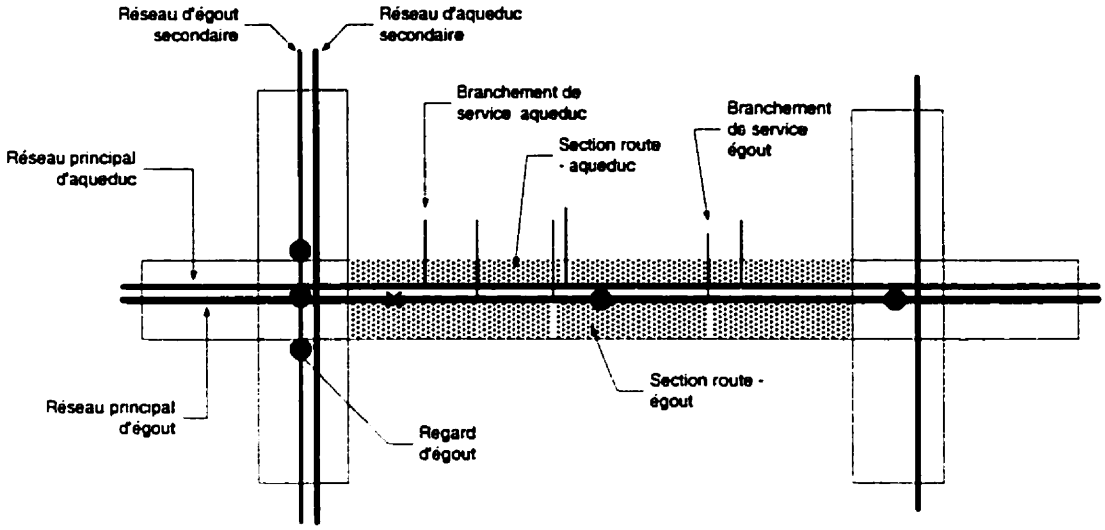


Figure 3. 9 Représentation d'une section

3.4.7 Package INVENTAIRE

Ce sous-système contient l'inventaire du système urbain. La distinction a été faite entre les éléments qui forment le réseau (les éléments de canalisation) et les autres éléments qui sont adjacents ou annexes. La Figure 3.10 montre un exemple de canalisation dans le cas du réseau d'assainissement. Tous deux peuvent être directement gérés ou non par la municipalité, et sont nécessaires à la modélisation complète de l'infrastructure urbaine. Parmi les éléments adjacents, on peut retrouver par exemple une station de pompage, une centrale électrique, un lac, etc. Ils peuvent être des éléments complexes qui pourront être modélisés afin de raffiner les performances des SIDEX et du SIGEC en général.

Parmi les éléments de canalisation, il faut faire la distinction entre les tronçons (ou lignes, ou segments) et les nœuds. En effet, la modélisation d'un réseau quelconque

se fait par ces deux éléments. On considère qu'un tronçon est toujours délimité par un nœud de départ et un nœud d'arrivée

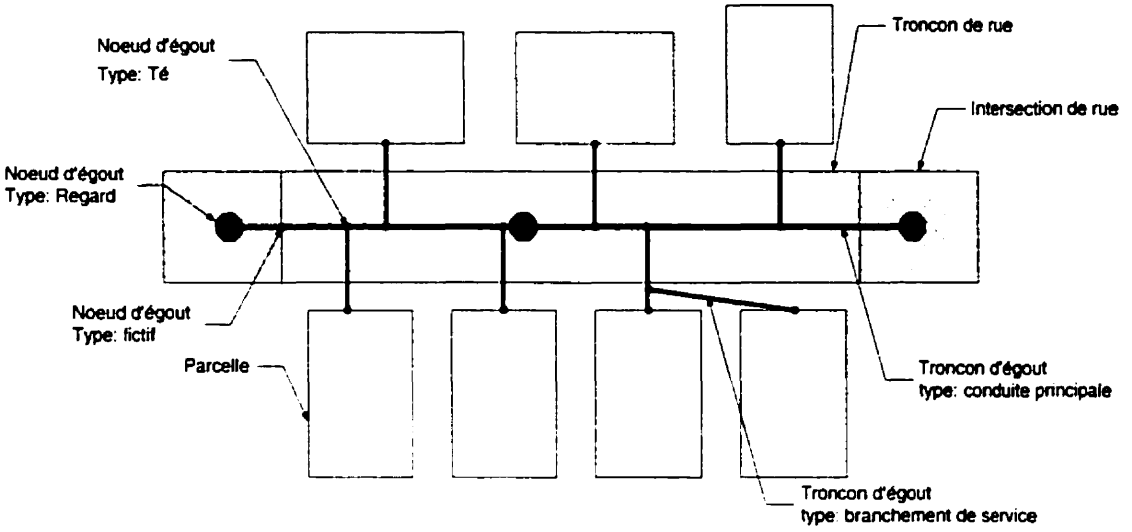
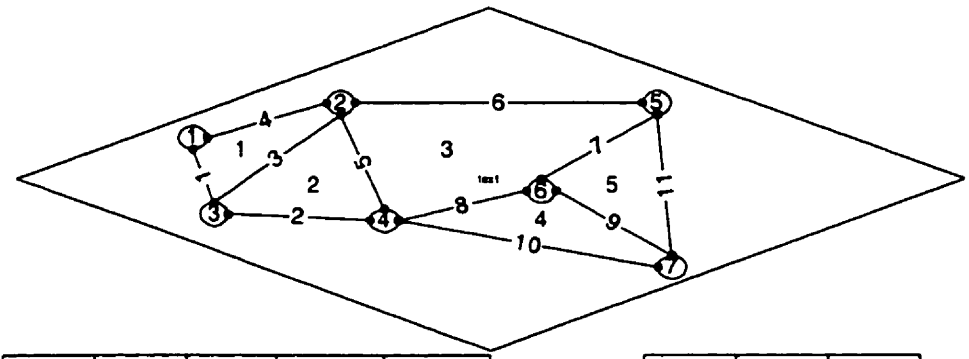


Figure 3.10 Représentation simplifiée d'un réseau d'assainissement

Un nœud peut être une borne d'incendie, une vanne, un « té ». Il peut également être fictif, pour souligner une démarcation à l'intérieur d'un tronçon (élargissement, changement de structure, ...), tandis qu'un tronçon de système urbain peut être une conduite principale ou secondaire, un branchement de service. La Figure 3.11 montre les relations entre les informations à références spatiales ainsi que la topologie d'un réseau avec une base de données.

Dans ce module, les classes *Sollicitation* et *ProprieteStructurale* caractérisent davantage les éléments du système urbain. Une sollicitation peut être définie comme l'origine d'une force physique (par exemple la pression) ou d'une réaction chimique (par exemple l'acidité) appliquée contre un ou plusieurs éléments du système urbain et qui, éventuellement, cause la détérioration de celui-ci (CERIU, 2000). Des exemples de sollicitations sont : pression, débit, eau, sol, coefficient de Hazen-Williams, trafic, gel, coefficient de Manning.



Ligne	Poly D	Poly G	Noeud 1	Noeud 2
1	1	0	3	1
2	2	0	4	3
3	2	1	3	2
4	1	0	1	2
5	3	2	4	2
6	3	0	2	5
7	5	3	5	6
8	4	3	6	4
9	5	4	7	6
10	4	0	7	4
11	0	5	5	7

Topologie

Noeud	X	Y
1	x1	y1
2	x2	y2
3	x3	y3
4	x4	y4
5	x5	y5
6	x6	y6
7	x7	y7

Coordonnées

Figure 3.11 Représentation d'un réseau quelconque

La classe *ProprieteStructurale* permet de conserver l'information sur la condition de l'élément du système urbain. Comme type d'indicateur, il existe : rouille, dépôt, quelconque (calcaire), déficience fonctionnelle (diamètre trop petit, route trop petite), dégradation chaussée (fissures), condition surface (charge supportée par la chaussée, sécurité), corrosion/érosion, infiltration, vitesse d'écoulement trop lente.

Dans le SIDEX générique, *Sollicitation* et *ProprieteStructurale* sont des classes abstraites. Lors de l'instanciation des autres SIDEX, ils pourront être spécialisés par type d'indicateur ou type de sollicitations. Ces ensembles de classes peuvent former des modules en soi. Dans le cas particulier du SIDEX chaussée, le module identification géographique n'a plus de raison d'être puisque l'identification même des éléments de la

route est implicite. La Figure 3.12 montre l'organisation des classes dans le « package » *INVENTAIRE*.

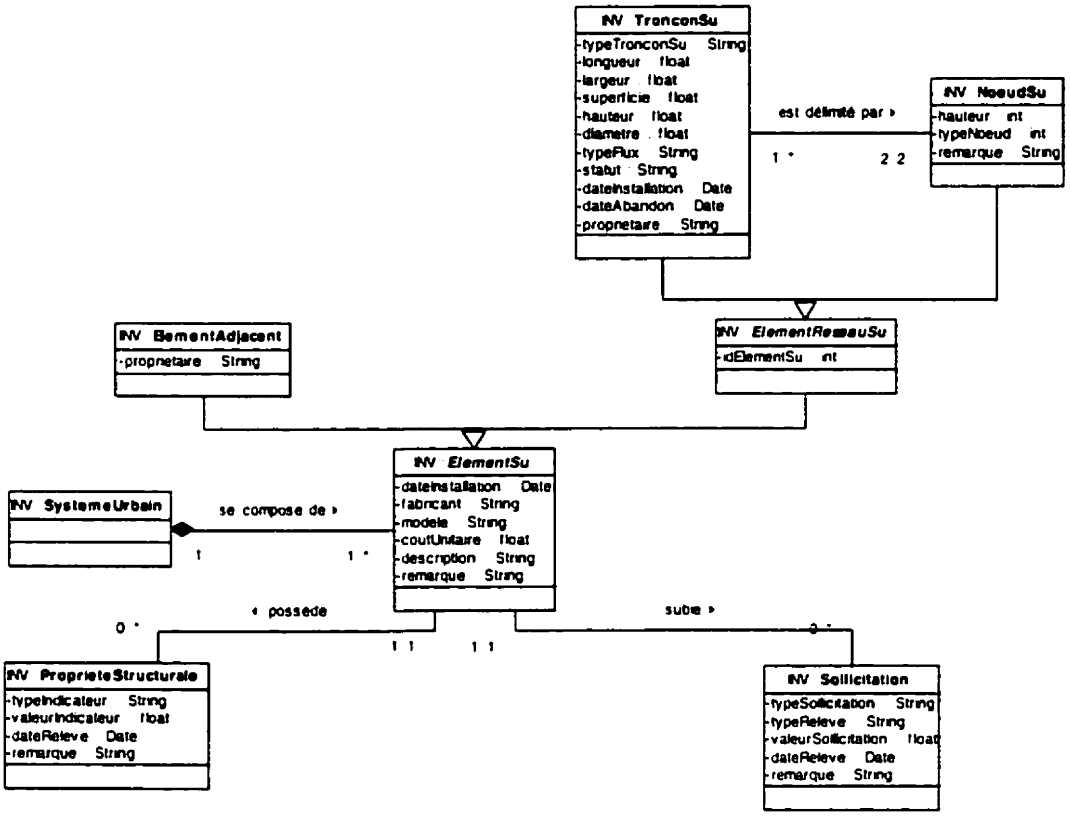


Figure 3.12 Package INVENTAIRE

3.4.8 Package MESURE

Ce sous-système contient l'information sur l'instrumentation qui est faite sur le système urbain. La Figure 3.13 montre l'organisation des classes dans ce « package ». Il contient des classes pour gérer l'aspect dynamique des SIDEX. Il ne contient pas une liste exhaustive de tous les équipements qui servent à mesurer les sollicitations ou les propriétés structurales des éléments du système. *Instrument* est une classe abstraite qui

contient les attributs minima qui permettent de caractériser un équipement de mesure, un API ou un équipement vidéo. Tous les éléments de ce module forment une structure complexe qu'il n'est pas essentiel de développer en détail. La classe Instrument possède une interface, une série de méthodes qui ne peuvent être définies de manière plus explicite à ce stade de développement du SIDEX générique. Ces interfaces accessibles directement sur l'instrument de mesure ou via un réseau viennent avec l'instrument. Certains instruments génèrent des relevés à un intervalle de temps défini. Ces relevés sont pris en compte dans le système et peuvent déclencher des alarmes selon le niveau d'un indicateur donné.

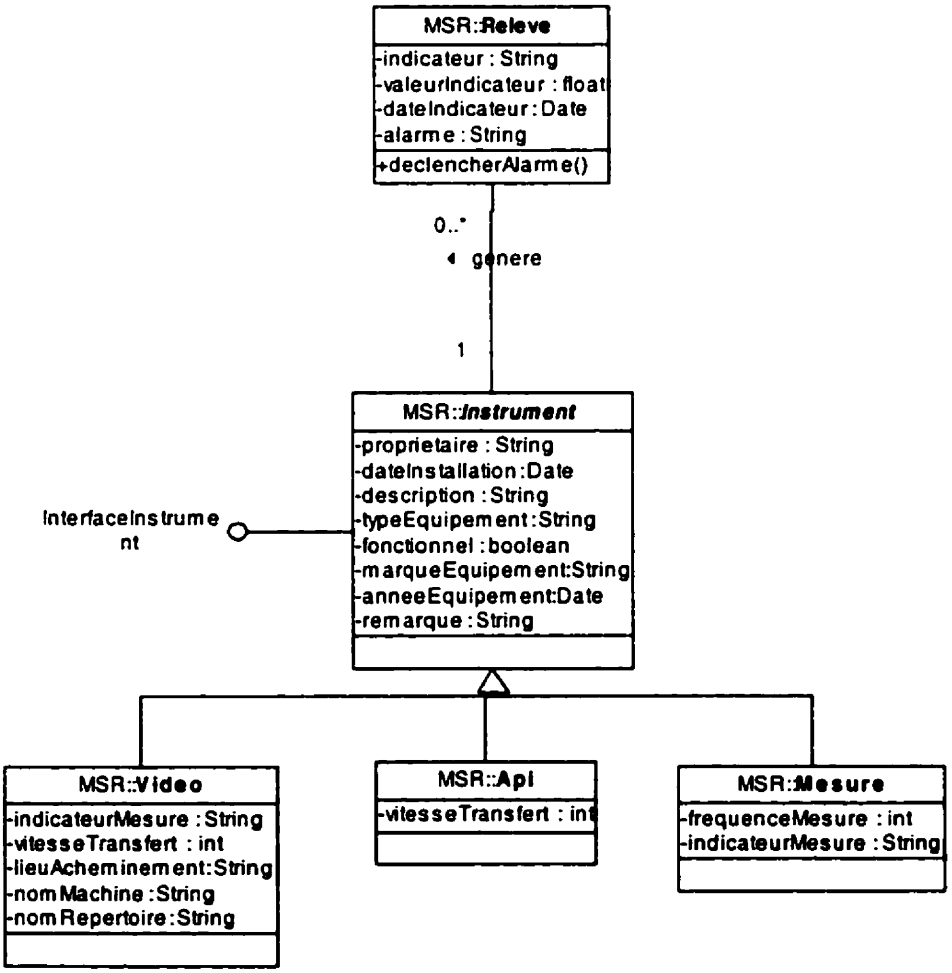


Figure 3.13 Package MESURE

3.4.9 Package TRAVAUX

Ce sous-système s'occupe de la gestion de tous les travaux de construction, d'entretien, de réhabilitation sur un système urbain donné. La Figure 3.14 montre l'organisation des classes dans ce « package ». Toutes les interventions, même mineures, sont gérées par les classes *Intervention*, *Personnel* et *EquipementTravaux*. De plus, les projets, les scénarios d'intervention et les programmes d'intervention peuvent être assignés à des travaux pour permettre une meilleure exécution ou coordination des tâches à effectuer. Comme pour le package *PLAINTE*, ce package pourrait se situer au niveau du SIGEC, car les travaux peuvent concerner plusieurs systèmes urbains.

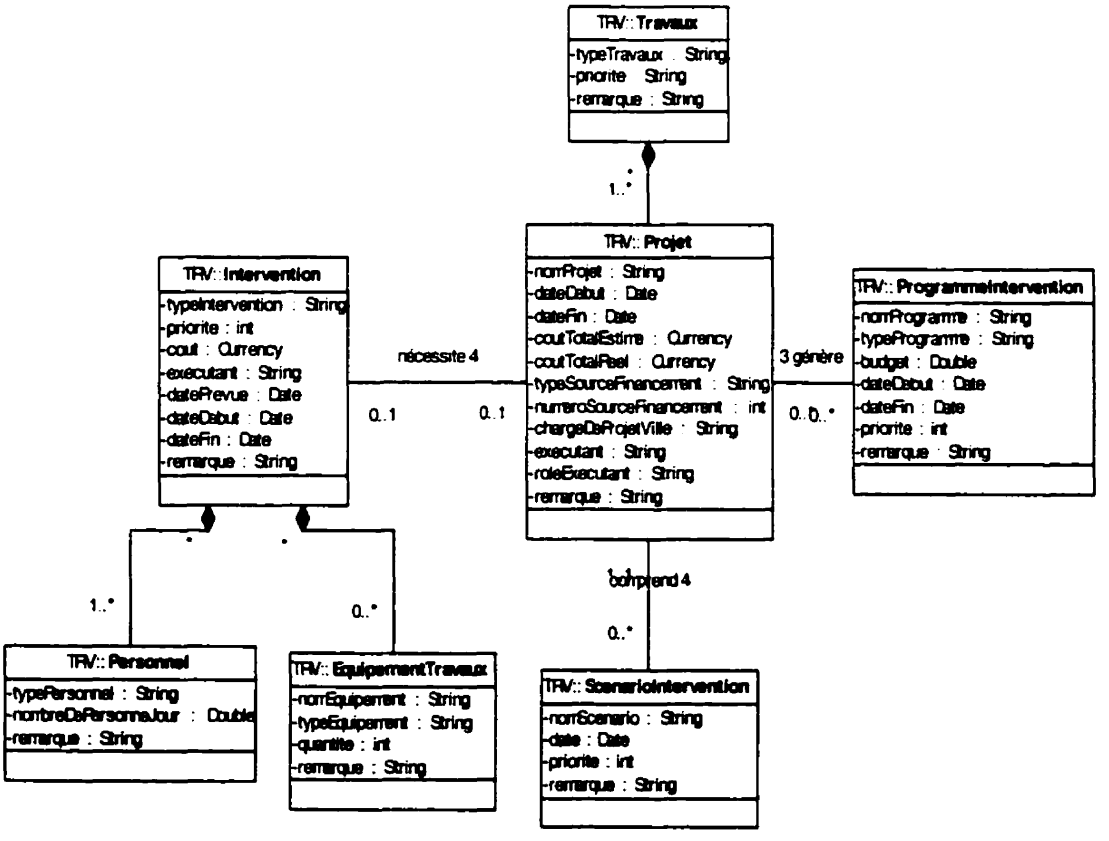


Figure 3.14 Package TRAVAUX

3.4.9 SIDEX générique

Le schéma conceptuel du SIDEX générique avec tous les « packages » est montré à la Figure 3.15. À remarquer qu'il y a 3 nouveaux « packages » qui ont été ajoutés. Parmi eux, le package *IMPACTENVIRONNEMENT* qui contient les classes nécessaires pour représenter l'impact des infrastructures urbaines dans le milieu naturel. Ce « package », comme les deux autres, est spécifié de cette manière afin que, lors de l'instanciation du SIDEX générique, on puisse en tenir compte dans les SIDEX spécifiques.

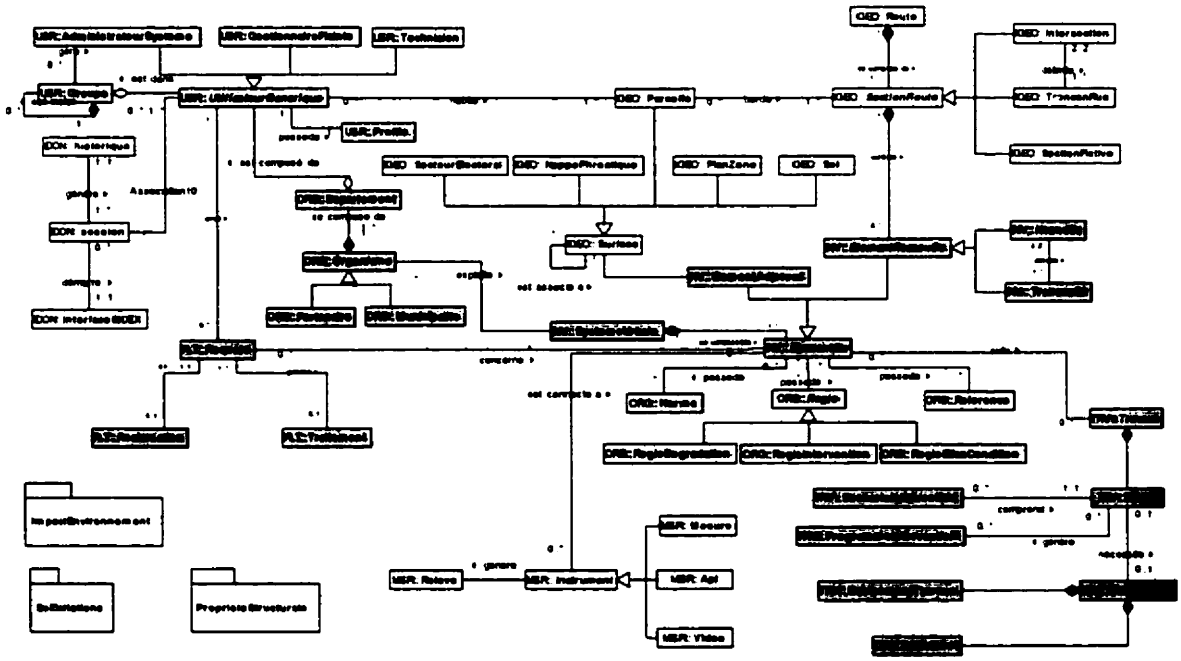


Figure 3.15 Architecture générale du SIDEX générique

3.5 Architecture informatique

L'étape précédente ayant présenté le modèle logique du SIDEX générique, il est important de définir l'architecture informatique sur laquelle se fera l'implémentation des SIDEX spécifiques. Le modèle proposé est une architecture client/serveur multicouche orientée web utilisant l'Internet avec les technologies qui gravitent autour. Ainsi, la solution est une combinaison de plusieurs architectures employées dans les applications d'aujourd'hui sur les réseaux privés des entreprises ou sur l'Internet. La Figure 3.16 montre les principales caractéristiques de cette architecture informatique. Comme dans les architectures vues précédemment au chapitre 2, elle se compose de clients (clients légers et clients lourds) répartis sur la première couche, de serveurs de bases de données réparties sur la dernière couche et finalement de serveurs d'applications réparties sur plusieurs couches entre les clients et les serveurs de bases de données. Entre les couches, les mécanismes de communication se déploient à travers le « middleware ».

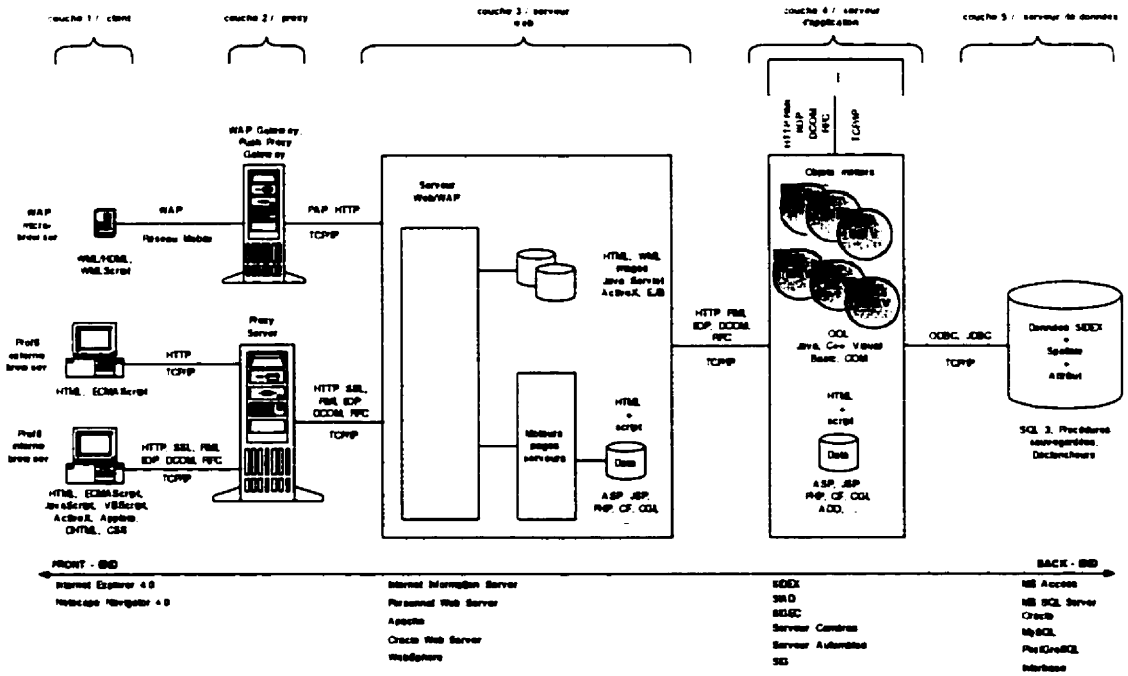


Figure 3.16 Architecture informatique du SIDEX générique

3.6 Mécanismes de surveillance et de communication

L'un des grands défis pour les municipalités est de favoriser une gestion dynamique ou en temps réel des infrastructures urbaines. Construites il y a plus d'un demi-siècle, celles-ci ont une durée de vie généralement assez longue lorsque leur entretien est systématique. De nos jours, avec les contraintes budgétaires, les municipalités ne peuvent se permettre d'ignorer l'évolution des technologies leur permettant de faire un suivi continu de leurs infrastructures à un coût moindre que de rebâtir en partie ou en totalité une infrastructure.

Un des volets de ce projet est d'étudier la faisabilité d'installer sur certaines de ces infrastructures des automates et des équipements de télécommunications pour la saisie, le traitement et la transmission de données destinées à la prise de décision éclairée. Les systèmes urbains sont répartis sur toute l'étendue géographique de la municipalité; il est donc nécessaire de disposer de réseaux de communications adéquats qui interconnectent ces systèmes urbains. Mais, avant de discuter des éléments de communication qui interviennent au niveau d'un SIDEX, il convient de savoir quelle information va être véhiculée au travers de ces réseaux. D'où la nécessité de savoir ce qui va être surveillé afin d'être acheminé au SIDEX pour analyse.

3.6.1 Mécanismes de surveillance

Les gestionnaires de première ligne à la municipalité désirent surveiller certaines parties des infrastructures urbaines qui, si elles ne le sont pas, peuvent être la cause de dysfonctionnement grave dans le réseau. La connaissance des conditions réelles de fonctionnement des systèmes urbains est primordiale à une prise de décision éclairée et passe très souvent par une surveillance en temps réel ou différé. D'où l'idée de gestion dynamique. L'idée d'un suivi continu des infrastructures est très bonne, mais il ne faut pas en abuser. Tout dépend du temps critique pour pouvoir prendre une décision à un point critique du réseau. Une municipalité ne va pas instrumenter le réseau électrique en entier afin de détecter les ampoules qui sont grillées sur une avenue. En effet, seuls les

points critiques sont surveillés, ceux susceptibles de représenter les conditions du système urbain en ce point et permettant d'interpoler pour avoir une condition plus globale, et également ceux susceptibles de causer des problèmes sur une partie de l'infrastructure. Aujourd'hui, la technologie est au rendez-vous des gestionnaires. Plusieurs types d'équipements peuvent mesurer la condition du réseau en des points critiques. Si l'on prend le débit dans une canalisation, une caméra ou une sonde ferait le même travail. Chacun de ces équipements a sa particularité et possède des avantages certains sur l'autre, mais la justification de l'installation d'un équipement sur une infrastructure dépend des lieux de mesure. Les éléments à prendre en compte lors de la prise de décision pour l'installation d'un équipement sont : le coût, la vitesse de transfert, le degré de précision, la maintenance ou la durée de vie dans l'environnement du système urbain.

3.6.2 Mécanismes de communication

Afin de véhiculer les données qui sont générées par la surveillance des infrastructures urbaines, il est important de bénéficier de réseaux de communication adéquats opérant sur toute l'étendue de l'infrastructure. Ces données sont de type multimédia : du texte, des images, de la vidéo, et du son. Il existe deux mécanismes de communication dans les SIDEX : les mécanismes de communication internes et les mécanismes de communication externes.

Les mécanismes de communication internes impliquent une communication au sein de l'application, c'est-à-dire inter-SIDEX et inter-SIGEC. Ces mécanismes font appel aux différentes méthodes de communication entre applications reposant sur les réseaux de communication utilisant la pile de protocoles TCP/IP. Ils sont développés lors de l'implémentation, entre les différentes composantes de l'application qui reposent sur un environnement réparti. Ce point important dans le processus de développement du SIDEX générique sera abordé dans le chapitre suivant lors de l'implémentation.

Les mécanismes de communication externes impliquent une communication qui se passe à l'extérieur du SIDEX. Généralement, elle implique des mécanismes de

communication entre les automates programmables industriels et les caméras, entre le SIDEX et les équipements de contrôle, entre le SIDEX et des utilisateurs mobiles.

L'aspect dynamique des SIDEX repose sur la présence d'équipements de mesure sur les infrastructures urbaines. Ces équipements de mesure sont des sondes, des pluviomètres, des débitmètres, des équipements capables de caractériser un flux de matière : eau sanitaire, eau potable, flux électrique, déplacement de voitures, ou autres. Ces équipements de mesure ne permettent pas de caractériser l'aspect statique des infrastructures, telle l'épaisseur d'une conduite qui grossit dû à la présence de dépôts. Cet aspect est déterminé par des inspections périodiques ou par la présence d'équipements vidéos. Ces instruments de mesure, bien que très souvent ne possédant pas d'écrans de lecture (ce qui ne serait d'aucune utilité dans bien des cas car les SIDEX désirent effectuer de la surveillance à distance), ont généralement un moyen de communiquer leurs données en temps réel à travers une interface de communication de type sériel.

D'un autre côté, des panneaux de contrôle opèrent sur les infrastructures afin de changer l'état structurel de l'infrastructure de manière automatique, c'est-à-dire fermer automatiquement une vanne dans le cas du réseau d'aqueduc ou sanitaire, éteindre des lumières dans le cas du réseau électrique, actionner une alarme dans le cas d'un bâtiment et bien d'autres possibilités. Ainsi, en plus de la mesure à distance (pour la surveillance), le contrôle à distance en temps réel est un des objectifs recherchés par les SIDEX. L'utilisation des automates programmables industriels est un élément indispensable pour effectuer ces tâches de contrôle.

Un automate programmable industriel possède généralement les caractéristiques suivantes :

- un ordinateur industriel renforcé pour fonctionner dans l'environnement généralement rigoureux des systèmes urbains ;
- une programmation par échelle classique ;
- un contrôle des entrées / sorties par la programmation utilisateur ;

- un jeu d'instructions spécialement élaboré pour le contrôle industriel et l'environnement de procédé ;
- une possibilité de communiquer avec des contrôleurs de cellules, des terminaux opérateurs intelligents (OIT), des terminaux « non intelligents », des ordinateurs personnels et d'autres équipements du même type.

Cet équipement répond parfaitement aux fonctionnalités de surveillance et de contrôle à distance recherchées par les SIDEX. En effet, un automate programmable industriel est un mini-ordinateur industriel possédant une unité centrale ainsi que plusieurs modules extensibles sur une platine (bac). Les équipements de mesure qui envoient de l'information sous forme d'impulsions, de caractères ou autres, ainsi que les équipements de contrôle, sont connectés à des modules spécialisés d'entrées/sorties (entrées logiques, sorties logiques, entrées analogiques, sorties analogiques et modules optionnels). Ces modules sont surveillés par l'unité centrale de l'automate programmable industriel ou par un ordinateur personnel (PC) possédant une carte d'interface adaptée. La Figure 3.17 montre un automate en action sur un système urbain. L'ordinateur hôte possède les outils de communication vers les SIDEX.

La communication entre un ordinateur personnel et un automate ou entre deux automates est de type sériel, utilisant des protocoles propriétaires. La Figure 3.18 montre les détails de communication en utilisant le principe de couche tel que défini par le modèle OSI ou la suite de protocoles TCP/IP. L'interface série code les caractères des messages, fournit la transmission électrique des messages sur la liaison série et contrôle la parité des caractères entrants. Le protocole de communication définit les règles d'établissement et d'interruption des liaisons pour l'envoi des messages et le transfert des données. Ils sont généralement propriétaires et ne peuvent donc être implémentés par les SIDEX. L'interface d'application API est l'interface de programmation qui permet d'établir les communications série sur l'automate.

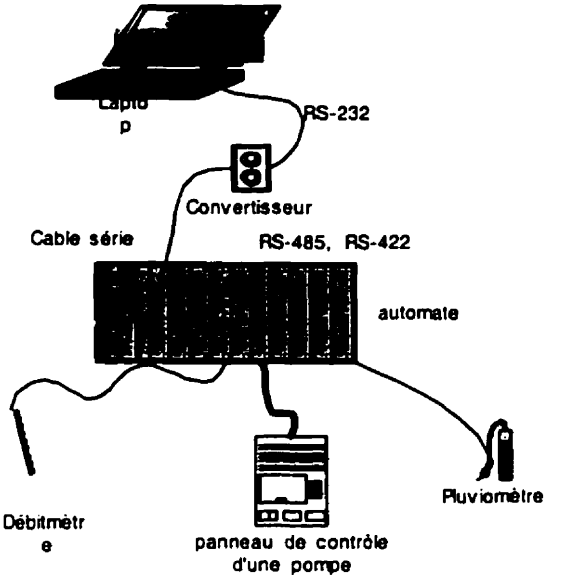


Figure 3.17 Connexion entre un automate et un ordinateur

Généralement, au niveau de l'interface d'application, on retrouve des logiciels propriétaires qui facilitent la programmation. Il est possible de créer un programme en Visual Basic ou en Java pour appeler des procédures (par exemple des fichiers dll dans le cas d'un système d'exploitation Windows) du logiciel.

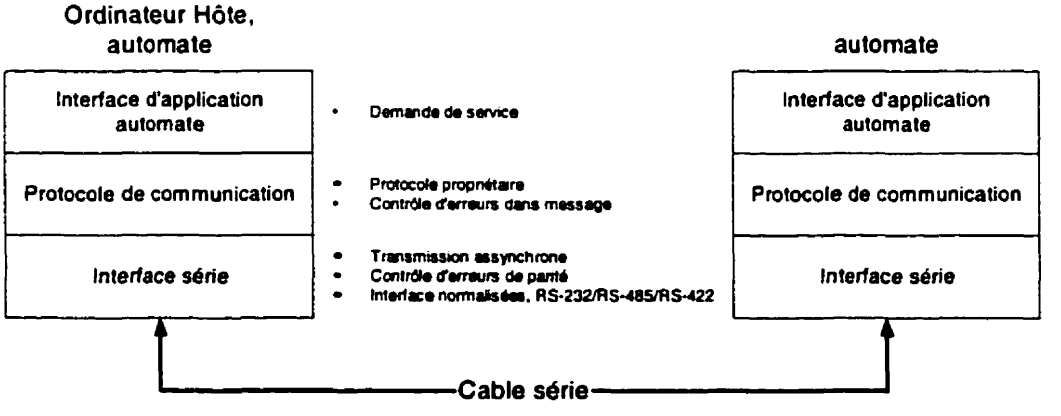


Figure 3.18 Principes de communication entre API et Hôte

La présence d'un mini-ordinateur a un impact décisif sur la centralisation des équipements de mesure ou de contrôle, ainsi que sur l'acheminement des données jusqu'au SIDEX. Car, considérer chaque équipement de mesure ou de contrôle avec son propre équipement de communication serait excessivement coûteux tant du point de vue matériel que du point de vue de la coordination. De plus, lorsque l'on prend une station de pompage, une centrale électrique, un bâtiment ou autre, la présence d'équipements de mesure ou de contrôle sera importante, d'où la nécessité de confier la gestion locale, l'acheminement des données vers le SIDEX à un réseau d'automates. Les SIDEX se trouvent sur le PC qui interagit directement avec le ou les automates via une communication sérielle. Or, cette communication sérielle, même en utilisant des répéteurs, a une distance maximale à respecter pour éviter un dysfonctionnement. De plus, rares seront les cas où le lieu d'hébergement des SIDEX et le lieu de mesure seront proches. Il faut alors utiliser les réseaux de communications qui facilitent le transport de l'information bien au-delà des limites des standards RS-232, RS-485 ou RS-422.

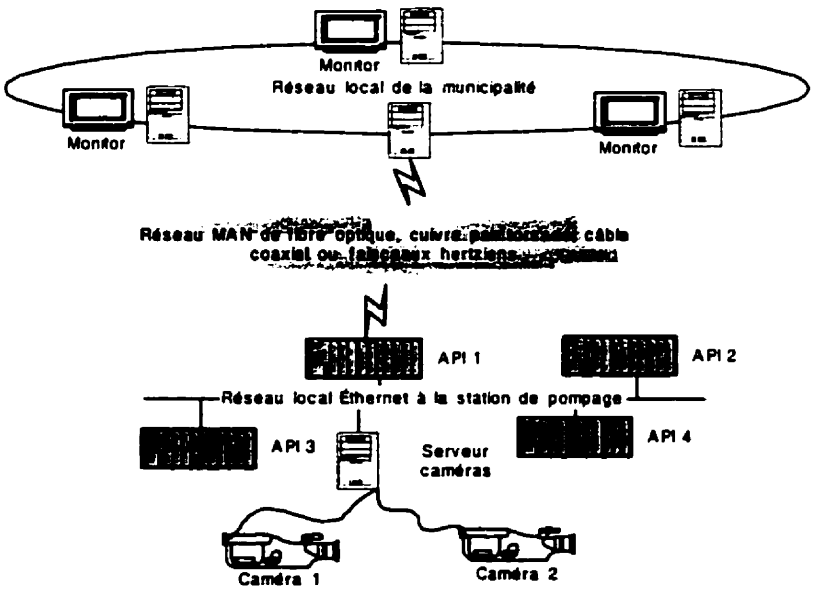


Figure 3.19 Schéma de connexion entre les SIDEX et les automates

La Figure 3.19 montre le schéma de connexion entre les SIDEX se trouvant à la municipalité et les automates se trouvant sur les infrastructures urbaines d'une municipalité. Dans ces cas, les principes de communication entre API et hôtes vont changer. La couche d'interface d'application ne change pas. Cependant, les couches qui fournissent des services vont changer. En effet, des standards ouverts sont utilisés afin de véhiculer l'information. Au niveau de la couche physique et de la couche liaison, les protocoles *Token Ring* et *Éthernet* peuvent être utilisés tant du côté de la municipalité que du côté des API opérant sur les infrastructures urbaines. La couche *réseau* utilise le protocole IP, tandis que la couche *transport* utilise le protocole TCP ou UDP selon l'application et les données transmises. En effet, si des images en temps réel sont transmises via le réseau ou les données ne sont pas d'une grande importance, UDP qui est moins fiable mais plus rapide serait préférable à TCP. Le réseau de communications métropolitain (MAN) entre les deux réseaux locaux peut être public ou privé. Dans ce cas, il va falloir installer des interfaces entre les protocoles *Éthernet* ou *Token Ring* et ceux utilisés par le MAN (ISDN, Frame Relay, ADSL). Dans le cas où les réseaux de données seraient excessivement coûteux à cause des données temps réel qui sont sans cesse demandées par les SIDEX, le réseau téléphonique commuté (PSTN) pourrait alors être utilisé. La Figure 3.20 montre le schéma de connexion lorsque les données passent par le réseau téléphonique. Les modems sont alors utilisés avec une connexion permanente ou temporaire pour le transfert de données. Cette méthode serait très efficace si les SIDEX avaient besoin de données pendant un intervalle de temps assez large.

C'est au niveau de la performance qu'il pourrait y avoir beaucoup d'inconvénients. Les réseaux locaux utilisant un protocole tel *Éthernet* fonctionneront à une vitesse de 10 ou 100 Mbps. C'est au niveau du sous-réseau de communication qu'il faudra faire par la suite des études de performance pour voir si l'application qu'est le SIDEX répond de manière efficace au temps réel.

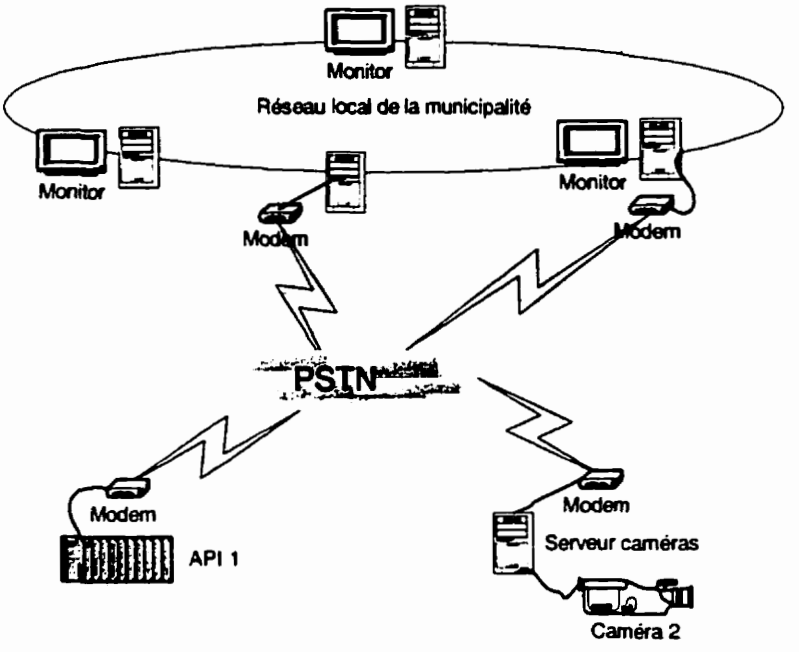


Figure 3.20 Schéma de connexion utilisant un réseau téléphonique

Par exemple, un événement comme celui-ci se déroule: le niveau d'eau augmente rapidement dans la station de pompage du réseau d'assainissement. Avec les capteurs (débitmètre, caméras) qui opèrent sur le système urbain, y aura-t-il assez de temps pour envoyer l'information vers le SIDEX avec le minimum de pertes, suivi d'une analyse de l'information puis d'une prise de décision assistée ou non, et finalement prendre une action comme la fermeture d'une vanne. Bien sûr, plusieurs critères doivent être considérés.

D'autres types de communication peuvent être considérés : communications sans fil mobile avec des usagers du SIDEX, communications fixes entre des équipements de mesure et des automates ou des ordinateurs hôtes. Dans tous les cas, les avancées

importantes de la technologie dans ce domaine répondent à la demande. Les technologies reposant sur le WAP seront nécessaires pour parfaire le système.

CHAPITRE IV
INSTANCIATION DU SIDEX GÉNÉRIQUE ET IMPLÉMENTATION

Le SIDEX générique est une abstraction empruntée aux concepts orientés objets et appliquée aux systèmes urbains. Une phase d'instanciation est donc nécessaire pour définir les caractéristiques qui sont propres à chaque système urbain. Cette étape permet de valider, d'adapter et de renforcer le modèle conceptuel générique décrit dans le chapitre précédent, suivant ainsi une démarche itérative. Ce présent chapitre traite de l'instanciation du SIDEX Égout et des méthodes d'implémentation des fonctionnalités propres à ce SIDEX. Cette phase d'implémentation conduira à la réalisation d'un premier prototype dédié au système urbain Égout.

4.1 Instanciation du SIDEX Égout

Les principes orientés objets ainsi que plusieurs spécifications UML sont appliqués tout au long de ce processus d'instanciation. UML étant un langage très ouvert, il permet de définir des stéréotypes représentant des extensions à la sémantique du langage. Ces derniers sont définis par des mots entre guillemets (« »). Le SIDEX générique est un « framework » qui, par extension du langage UML, constitue un « package » présentant une base de travail (template) pour être utilisé ultérieurement dans un domaine précis. Ainsi, le processus d'instanciation peut être considéré comme le déploiement de ce « framework » et la continuation d'un travail de modélisation en UML jusqu'à la définition complète du SIDEX Égout. Ce framework peut être modifié pour chaque instanciation. Les conditions d'extensions du SIDEX générique doit répondre au moins au critère de minimalité. Pour chaque élément à rajouter, il faut se

poser la question si cet élément apparaît dans chaque SIDEX spécifique lorsque le SIDEX générique est instancié.

Le modèle conceptuel de donnée du SIDEX générique sert de référence. En effet, les SIDEX spécifiques sont des applications de ce « framework », à la manière d'un héritage simple du concept orienté objet. Il n'est pas nécessaire d'effectuer ici un travail rigoureux d'analyse comme cela se fait généralement en modélisation de système ou de processus. Car le but du SIDEX générique, en plus de définir une structure de données normalisée à travers la dimension statique du modèle objet, est de minimiser le temps, ainsi que les coûts de conception et de déploiement d'un SIDEX en particulier. Il est donc tout à fait normal que l'analyse effectuée dans le cadre du SIDEX générique ne soit pas répétée systématiquement pour chaque SIDEX spécifique, mais plutôt vérifiée pour renforcer le modèle générique.

L'analyse et la conception du SIDEX Égout ont donc porté sur les éléments propres à l'Égout qui n'apparaissent pas dans le SIDEX générique. Les nouvelles classes qui apparaissent dans le module système urbain découlent d'une analyse rigoureuse basée sur les diagrammes de scénarios que l'on retrouve dans le SIDEX Égout. De plus, les éléments nécessaires sont ajoutés pour implémenter la gestion des plaintes relative à l'égout.

Dans le cas du système urbain Égout, seul le « package » *Inventaire* du SIDEX générique a subi des changements majeurs. En effet, un réseau d'égout ou d'assainissement se compose essentiellement de nœuds et de conduites. Les conduites n'ont besoin d'être rattachées qu'à une seule classe : *conduite*. Les nœuds se spécialisent en des éléments qui caractérisent un réseau d'égout en général. Les autres classes dans les autres « packages » n'ont subi pratiquement aucun changement car les classes qu'ils contiennent sont très représentatives des éléments que l'on retrouve dans un système intégré d'exploitation. Le modèle logique du SIDEX Égout est présenté à l'Annexe C. L'instanciation du SIDEX Égout est effectuée dans le but de créer un prototype qui va valider le SIDEX générique. Quelques classes pour l'implémentation du prototype axé sur la gestion des plaintes sur le réseau d'assainissement ont été rajoutées. Ce sont celles

qui sont définies dans le « package » *InterfaceDonnees*. Ces nouvelles classes ont été ajoutées lors de la deuxième itération du SIDEX générique présenté au chapitre précédent. Cette étape permet d'étendre le SIDEX générique. La Figure 4.1 présente le « package » *Inventaire* du SIDEX Égout.

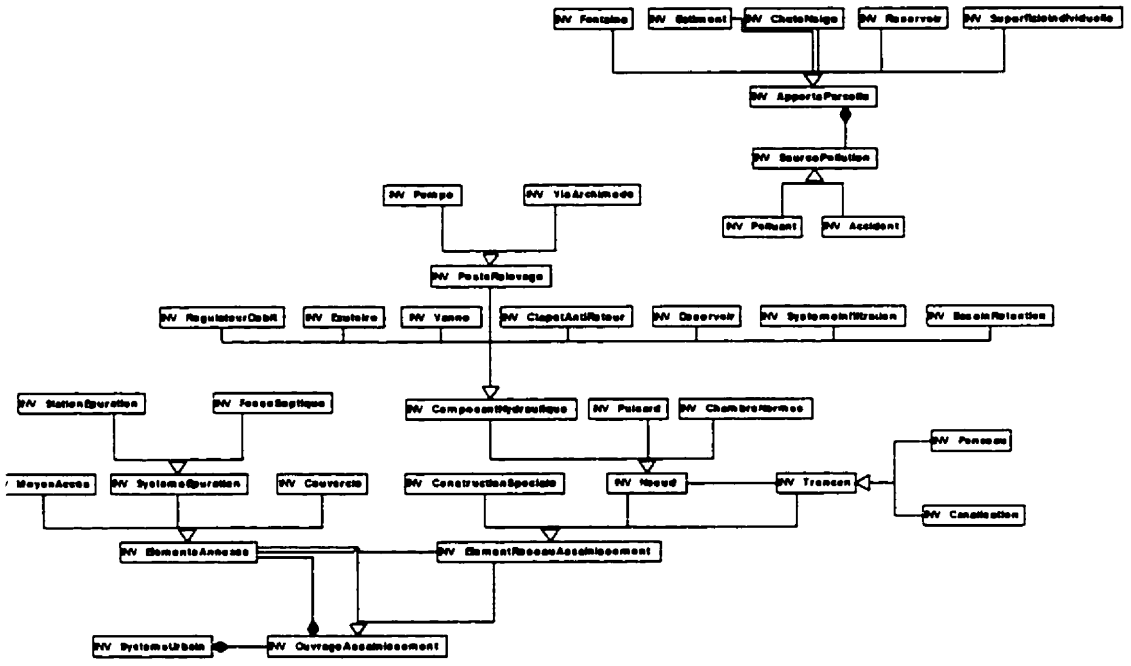


Figure 4.1 Package Inventaire du SIDEX Égout

4.2 Processus d'instanciation

Lorsqu'on veut passer à l'implémentation, les considérations matérielles et logicielles exigent quelques restrictions. En effet, selon la base de données (relationnelle ou objet), il faudra tenir compte de la séparation entre les données et le traitement. De plus, selon le nombre de SIDEX instancié, on peut avoir plusieurs « packages » instanciés ou non dans le SIDEX ou dans le SIGEC. En effet, prenons l'exemple du

module de gestion des plaintes du SIDEX Égout, les plaintes sont générales et peuvent concerner plusieurs systèmes urbains simultanément. Il se peut que la manière de représenter et de gérer les plaintes dans ce SIDEX ne soit pas similaire au traitement dans les autres SIDEX. Alors, il faut redéfinir la structure de données ainsi que les objets propres au module *PLAINTE* du SIDEX Égout. Cela revient à rafiner le module plainte lors de l'instanciation. Venu le moment de l'implémentation, le SIGEC, grâce à son module de coordination, garde une référence à cet objet *plainte* de telle manière qu'il soit accessible à tous les SIDEX concernés. Le reste de la structure de données propre au SIDEX Égout est maintenu au niveau du SIDEX. Dans l'exemple du prototype de développement des plaintes, on considère que la structure de données d'une plainte ainsi que les traitements associés sont similaires à tous les SIDEX instanciés. Ainsi, avec un seul SIDEX instancié, les plaintes sont gérées en totalité par le SIGEC tel que montré à la Figure 4.2.

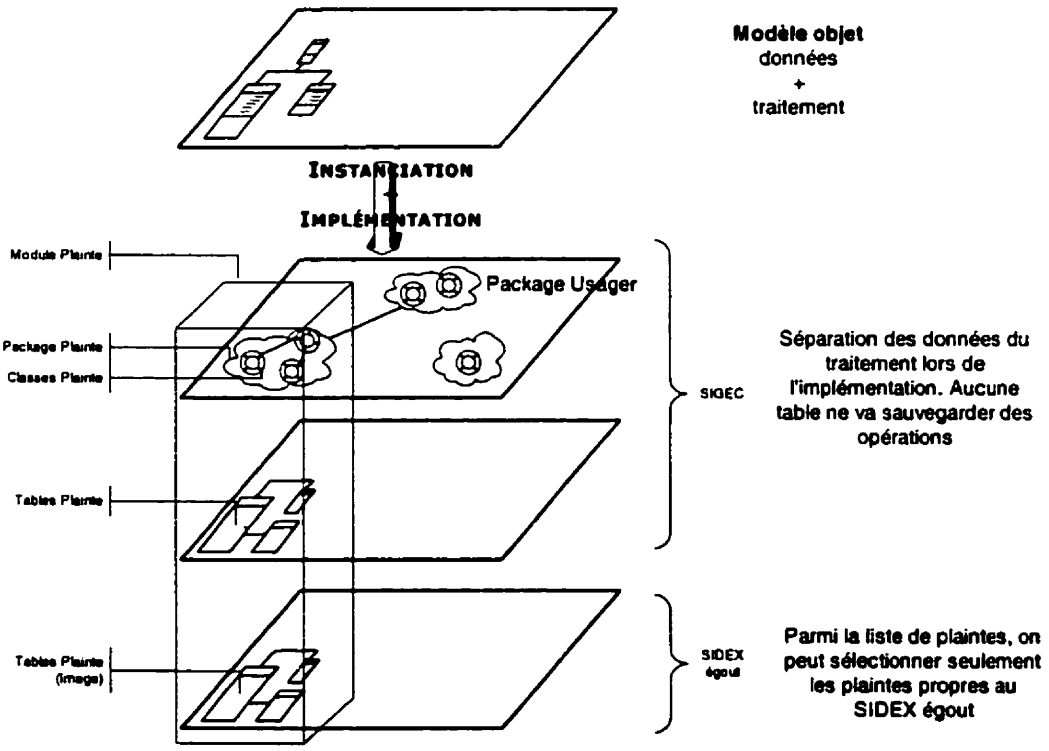


Figure 4.2 Séparation traitement et données

4.3 Considérations sur le choix de la méthode d'implémentation

Afin d'illustrer la conception de systèmes d'exploitation pour infrastructures urbaines par instanciation du SIDEX générique, un prototype pour le SIDEX Égout est conçu. Dans cette section, nous nous attarderons sur l'environnement informatique de développement et sur les détails d'implémentation d'un prototype pour la gestion des plaintes dans le SIDEX Égout.

4.3.1 Environnement matériel et informatique de développement

Dans cette section, nous montrons les éléments matériels sur lesquels repose le développement du prototype pour la gestion des plaintes du SIDEX Égout. L'application étant répartie sur plusieurs couches, il est normal de définir le matériel sur chacune de ces couches. Le prototype a été développé sur deux couches matérielles bien qu'étant une architecture 3 tiers du point de vue informatique. La base de données ainsi que les composants métiers qui encapsulent la logique de l'application se trouvent donc sur la même machine. Cette dernière possède les caractéristiques suivantes:

Processeur: Pentium III 300 MHz ;

Mémoire vive (RAM): 128 Mb (minimum) ;

Système d'exploitation: Windows NT Workstation 4.0 avec SP5 et Option Pack ;

Disque dur: 15 GB ;

Serveur web: Internet Information Server 4.

Bien sûr, une machine ayant des caractéristiques similaires avec des capacités supérieures peut également faire l'affaire. Par ailleurs, si l'on veut répartir l'application sur plusieurs couches afin d'augmenter les performances de notre application en général, il faudrait disposer de machines ayant les caractéristiques similaires, mais dotées de cartes réseau. Les caractéristiques de la carte seraient de préférence Ethernet 10 Mb, 100 Mb ou 1 Gb. La séparation du serveur web en tant que frontal ainsi que des composants

métiers est pratique courante dans l'industrie. Pour la configuration du client, elle a été testée sur des machines avec les caractéristiques suivantes:

Navigateur web: Internet Explorer ou Netscape Navigator 4.0 et plus ;

Modem: 28.8 kbps et plus; ou carte réseau pour accéder à l'Internet.

L'environnement informatique sur lequel reposent ces applications est très hétérogène. Il fait intervenir aussi bien les systèmes de gestion de bases de données que les interfaces utilisateurs en passant par des serveurs d'applications. L'environnement de développement devrait être en conséquence.

Il est rare de trouver des outils CASE qui prennent en compte tous le processus de développement d'un système client/serveur. Par exemple, *Rational Rose 2000* et *Visio 2000* sont des outils de modélisation très puissants, capables de faire de la conception et une partie de l'implémentation (possibilité de générer du code en Java, VisualBasic, C++ et autres). On est donc obligé d'utiliser plusieurs environnements de développement pour des étapes bien précises du processus de développement d'un logiciel. Ainsi, *Visio 2000 Enterprise* ainsi que *Dreamweaver 3.0* ont été utilisés pour le prototype.

Au niveau des bases de données, le SIGEC utilise la base de données *PostgreSQL* dans l'environnement Linux pour la gestion des suggestions dans le module des Plaintes. La base de données *Microsoft Access 2000* sous Windows est également utilisée pour le SIDEX Égout ainsi que pour le module des plaintes. De plus, *Microsoft Access*, bien que très lourd pour la gestion de grandes quantités de données, a été choisi à cause de sa facile intégration avec le logiciel de modélisation *Visio 2000*. Le temps de développement se trouve grandement diminué car l'implémentation du modèle logique (diagramme conceptuel) en un modèle physique (table dans Access) est un processus automatisé dans *Visio 2000*. Ce dernier possède la possibilité de faire du « reverse engineering » en maintenant le modèle logique à partir du modèle physique.

Les utilisateurs accèdent au SIGEC ainsi qu'aux SIDEX à travers une interface web. C'est l'interface idéale pour le partage et la diffusion de l'information à grande échelle. Les développements en cours tiennent comptent de l'hétérogénéité des fureteurs se conformant aux directives du W3C. Les langages suivants sont utilisés pour la conception de ces interfaces : HTML 4.0, CSS 2.0, JavaScript1.2 et DHTML. Les trois derniers langages ajoutent de la convivialité aux interfaces en permettant le positionnement absolu ou relatif d'objets dans une page, le formatage du texte ainsi que la programmation pour pouvoir réaliser le métier. Bien qu'une simple console texte suffise pour concevoir ces interfaces, nous avons utilisé l'environnement de développement du logiciel *Dreamweaver 3*.

Au niveau des serveurs d'applications, les langages de script PHP et ASP sont utilisés. Ces derniers sont des langages pseudo orientés objet en ce sens qu'ils n'implémentent pas tous les concepts orientés objet. Au niveau de PHP, l'héritage multiple n'est pas supporté (on ne peut hériter de plus d'une classe). Lorsqu'on possède une classe fille, l'appel du constructeur de la classe parent doit être explicite (implicite en C++). Il n'y a pas de destructeur en PHP. « L'overloading » et « l'overriding » ne sont pas supportés par PHP (concept très important qui économise beaucoup de lignes de code). Certains puristes diront que PHP n'est pas un langage purement orienté objet, ce qui est vrai. PHP est un langage hybride qui permet d'utiliser les concepts orientés objet avec la programmation traditionnelle procédurale. Une des grandes lacunes dans les langages de scripts actuels est la non persistance des objets tout au long d'une session. Deux techniques existent : soit la création d'objets qui sont compilés, soit le passage de paramètres d'un objet. La deuxième méthode a été utilisée car on reste dans le domaine des scripts, et en terme de développement rapide d'un prototype, il est plus facile d'utiliser les langages de scripts plutôt que du code compilé.

Dans le prototype de SIDEX, le serveur web représente un frontal pour le client. Par la suite, les requêtes sont distribuées aux serveurs d'applications concernés (fichiers scripts). C'est ainsi que le module des plaintes utilise des scripts PHP 4.0 installés en

tant que module CGI dans le serveur web IIS 4.0. Le système d'information à référence spatiale utilise des servlets java sur Apache et des scripts ASP sur IIS.

La prochaine étape consiste à mettre sous forme de code le modèle conceptuel du SIDEX Égout.

4.3.2 Interfaces utilisateurs

L'interface se compose essentiellement d'objets inclus dans le DOM. Ces objets ont différentes propriétés et méthodes. Il est possible de les créer directement avec du HTML ou dynamiquement avec JavaScript. La première option a été choisie car elle est plus pratique pour un développement rapide, tandis que la deuxième option n'est pas encore correctement standardisée. Le langage HTML est un langage de balisage pour formater le texte, il se compose de balises ou tag. Certaines de ces balises sont des objets en soi car ils sont définis dans le langage HTML et dans le DOM tel <select> <frame>, etc. D'autres tels ou ne sont pas des objets, ce sont de simples contenants non définis dans le DOM. De préférence, tout élément qui se trouve dans une page doit être modifiable par programmation. Cela nécessite que cet élément soit encadré par un tag défini dans le DOM.

Ces interfaces utilisateurs contiennent également des scripts (déclencheurs) pour la gestion des évènements locaux. Parmi les scripts développés, on retrouve deux grandes catégories : des scripts d'interaction avec l'utilisateur (validation de formulaire, déplacement de la souris, utilisation du clavier), des scripts d'interactions avec l'environnement informatique de l'usager (vérification de la taille de l'écran, vérification des fenêtres, vérification du type et de la version du fureteur, vérification et installation de composantes ou d'objets chez l'utilisateur). Ces déclencheurs sont très importants et permettent une interaction facile avec le client. Pour les déclencheurs de validation, ces derniers font un premier filtrage avant d'envoyer l'information sur le réseau, ce qui décongestionne ce dernier en cas d'erreurs pour un champ vide. La Figure 4.3 est un exemple de script utilisé dans les différentes interfaces développées pour la gestion des plaintes.

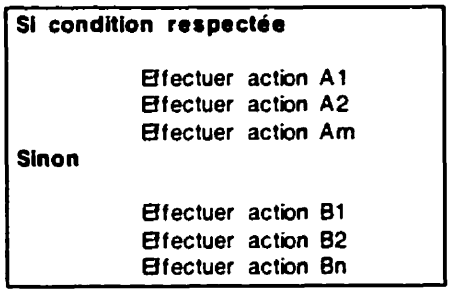


Figure 4.3 Script générique pour la validation de champs de formulaires

À noter que les actions Ai ou Bi peuvent également être des conditions à vérifier. L'instanciation de l'algorithme générique pour la validation du formulaire d'authentification des utilisateurs se fait comme indiqué à la Figure 4.4.

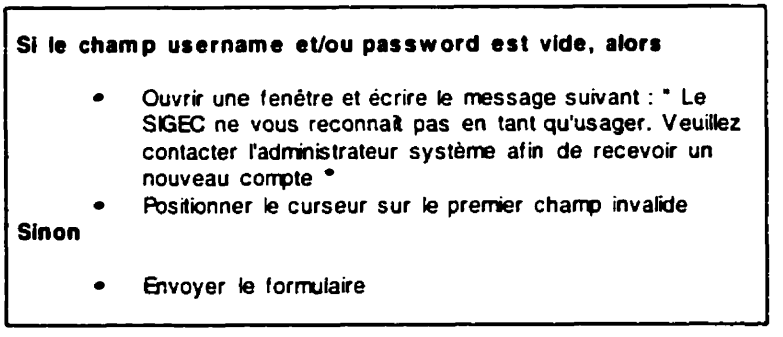


Figure 4.4 Instance de l'algorithme pour l'identification d'un usager

La Figure 4.5 présente un code en JavaScript permettant de valider la page d'entrée *default.php* avant la soumission du formulaire. Ce script est inclu dans l'entête de la page.

```
// validation du formulaire de login
function Validation(){
        erreur = false;

        if(document.forms[0].elements[0].value==""){
                erreur = true;
                if(document.forms[0].elements[1].value==""){
                        alert("Veuillez entrer votre nom d'usager et votre mot de passe");
                        document.forms[0].elements[0].focus();
                        return !erreur;
                }
                alert("Veuillez entrer votre nom d'usager");
                document.forms[0].elements[0].focus();
                return !erreur;
        }
        else if(document.forms[0].elements[1].value==""){
                erreur = true;
                alert("Veuillez entrer votre mot de passe");
                document.forms[0].elements[1].focus();
                return !erreur;
        }
        return !erreur;
}
```

Figure 4.5 Code javaScript de validation à la page d'entrée

L'exemple de script générique est également utilisé pour compléter automatiquement une partie de formulaire lors de la saisie d'une plainte. En effet, avant de saisir une plainte dans la base de données, il faut s'assurer que les nom, prénom et localisation des dommages sont rentrés par l'utilisateur. De plus, un script du même type est exécuté pour compléter automatiquement les lieux de dommage lorsque l'utilisateur spécifie, par une case à cocher, que ces lieux correspondent au domicile du plaignant. Bien d'autres scripts sont utilisés dans le module de gestion des plaintes pour améliorer la convivialité de l'interface et la sécurité de l'application. Cependant, la majorité d'entre eux est basée sur le principe de fonctionnement du script générique défini à la Figure 4.3.

4.3.3 Bases de données

Le modèle conceptuel de la base de données représente les concepts qui doivent être conservés dans une base de données. En effet, la base de données n'est pas le seul mécanisme de réalisation de la persistance. Par exemple, certaines données peuvent être maintenues comme des constantes du programme d'application. Le choix du mécanisme de réalisation de la persistance ne doit pas être déterminé à l'analyse mais lors de l'étape de la conception. Le choix des technologies employées pour l'implémentation de la persistance a une influence sur le modèle conceptuel de la base de données. Étant donné que le processus de développement du système est itératif, certains choix technologiques au niveau de la persistance sont fait implicitement.

La conception de l'application a un impact sur la conception de la base de données qui, à son tour, a un impact sur la logique et sur l'interface. Si la base de données ne contient pas l'information sur les dates des pluies par exemple, on ne pourra pas montrer l'historique des pluies à un usager. Il est important d'être rigoureux dans le passage du modèle conceptuel de l'application à celui de la base de données. Dans bien des cas, le diagramme de classe en UML correspond exactement au diagramme Entité-Relation aussi appelé Entité-Association. Le diagramme de classe UML est essentiellement une évolution du modèle entité-association adaptée à la modélisation objet. Des différences commencent à émerger lorsque l'on veut représenter les opérations et les relations entre les classes. Les classes abstraites définies dans le modèle générique du SIDEX n'existent pas dans le modèle relationnel ou objet relationnel, contrairement aux bases de données orientées objets. Les bases de données relationnelles possèdent déjà des données pour les classes abstraites. Ces données représentent les objets qui sont créés à partir des classes abstraites. Pour la conception orientée objet, il est préférable d'utiliser des attributs ayant des types de données primitifs qui sont offerts par la base de données. Par exemple, *Oracle 8* supporte les primitives suivantes *number*, *varchar*, *date*. Pour représenter des objets qui possèdent une structure non définie par les primitives de la base de données, l'utilisation des associations est alors nécessaire.

Une méthode est une implémentation concrète d'une opération. Lorsque l'on crée le schéma de la base de données relationnel, les attributs sont sauvegardés dans la base de données. Les méthodes ne sont pas sauvegardées puisque les tables n'ont pas d'état et ne sont pas des types. L'implémentation des méthodes se fait au niveau du serveur d'application. Les « triggers » ou gestionnaires d'évènements sont des procédures orientées tuples. En ce sens, ils sont déclenchés par des modifications apportées aux attributs d'une ligne.

Dans la base de données, chaque objet doit être identifié de manière unique. Dans le diagramme UML, ceci est défini de manière implicite, c'est-à-dire les classes ne possèdent pas un attribut qui définit son unicité. Par contre, dans le schéma de la base de données, cela est implémenté par une colonne ayant une clé primaire (tout nouvel enregistrement doit avoir une valeur différente dans la colonne qui représente la clé primaire).

La base de données conçue est une implémentation directe du modèle conceptuel de données du SIDEX Égout. Elle contient l'information relative à tous les modules nécessaires à l'exploitation du système urbain égout. Ces données sont de deux types : local et global. Au niveau local, on retrouve des données propres à l'égout qui ne sont utilisables que par le SIDEX Égout. Parmi ces données, il y a les données des modules *Inventaire, Mesures, Sollicitations, ProprietesStructurales.* Au niveau global, on retrouve les données qui sont partagées par les différents SIDEX, donc des données qui doivent se retrouver dans la base de données du SIGEC. Parmi ces données, il y a les données des modules *IdentificationGéographique, Travaux, Utilisateur, Plaintes, Organisation.*

La conception du schéma de la base de données relationnelle à partir du modèle UML fait appel à plusieurs techniques qui ne sont pas standardisées. Ils font appel au respect de certaines règles de base comme par exemple les formes normales jusqu'au niveau 3 au moins, et à l'utilisation des contraintes d'intégrité selon les règles de l'application. Il existe plusieurs méthodes pour l'implémentation. L'association binaire, la généralisation/spécialisation, l'agrégation/composition, l'auto-agrégation, sont des

exemples qui seront traités dans les quelques lignes qui suivent. Le diagramme entité-relation qui découle de la modélisation préliminaire en UML est présentée à l'Annexe D.

Associations binaires : suivant la sémantique d'UML, les associations binaires sont des associations qui lient deux classes. Ainsi, lors de l'implémentation, cette association se manifeste par une migration de clé primaire d'une table vers l'autre. Elle deviendra ainsi clé étrangère dans l'autre table. La multiplicité joue un rôle important dans la migration de la clé.

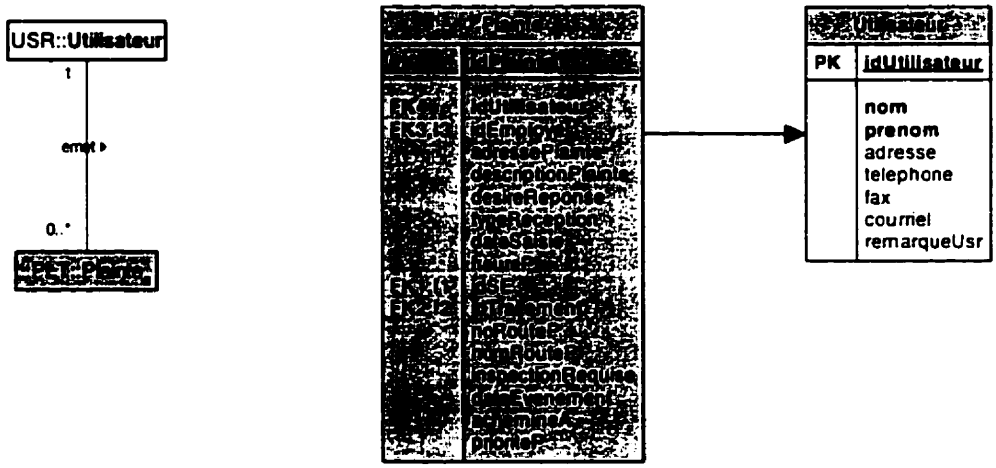


Figure 4.6 Implémentation d'une association

Dans l'exemple de la Figure 4.6, à gauche, on retrouve le modèle conceptuel du SIDEX Égout qui relie un utilisateur à zéro ou plusieurs plaintes. Dans la partie de droite, pour chaque plainte, on met l'identifiant de l'utilisateur dans la plainte.

Généralisation/spécialisation : UML définit ces deux concepts comme une relation entre une classe générale et une classe plus spécifique. La relation entre un utilisateur (général) et un employé (spécifique) est illustrée à la Figure 4.7.

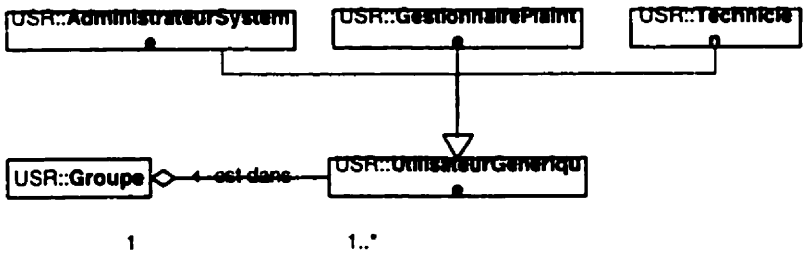


Figure 4.7 Relations entre utilisateur, employé et groupe

L'agrégation, illustrée à la Figure 4.7 et tel que défini par UML, est la partie qui forme le tout. Un utilisateur peut être un employé de type *AdministrateurSysteme, GestionnairePlainte* ou *Technicien*. De plus, un employé peut se trouver dans chacun de ces rôles différents, donc dans plusieurs groupes. Et dans chaque rôle, il n'appartient qu'à un seul groupe. Comme il peut avoir plusieurs rôles, il peut appartenir à plusieurs groupes différents. Un utilisateur est obligé d'appartenir à au moins un groupe.

Selon la Figure 4.8, pour illustrer une relation de spécialisation, on ne fait qu'une migration de clé de la table générale vers la table spécialisée. La relation d'agrégation des utilisateurs dans les groupes ainsi que la possibilité d'appartenir à plusieurs groupes est représentée par la création d'une table supplémentaire appelée *GroupeEmploye*.

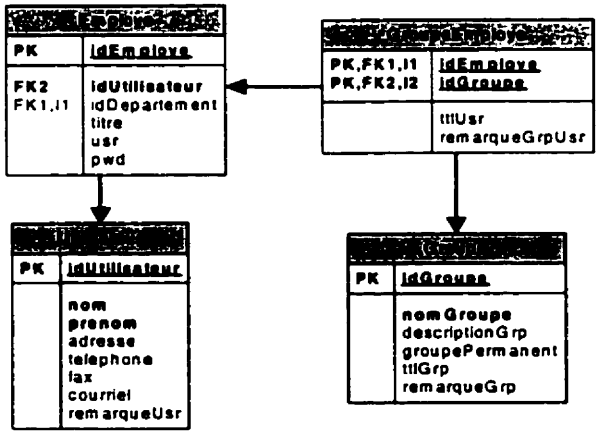


Figure 4.8 Relations de spécialisation et d'agrégation

L'auto-composition ou l'auto-agrégation est le fait que les objets d'une classe se compose des objets de la même classe. La Figure 4.9 montre un groupe qui est inclu dans un autre groupe. Le système doit être suffisamment flexible pour permettre de créer des groupes autres que ceux définies par défaut.

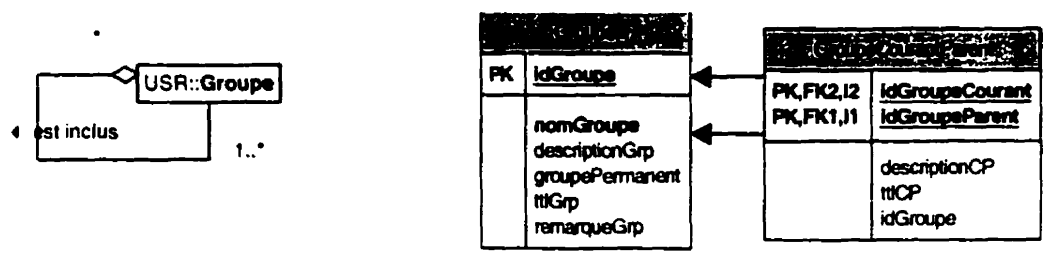


Figure 4.9 Création d'un groupe par auto-composition

La relation entre groupes est définie par la double relation qu'il y a entre la table *Groupe* et la table *GroupeCourantParent*. En effet, on est obligé de faire migrer deux fois la clé primaire groupe, d'abord pour représenter le groupe courant et ensuite pour représenter tous les groupes qui sont parents à ce dernier.

L'implémentation de la majeure partie du SIDEX Égout se fait par l'implémentation de la structure de persistance et cette dernière suit les règles définies ci-dessus pour passer directement d'un modèle conceptuel vers un schéma physique de la structure de persistance (SQL). Les tables et les colonnes sont structurées d'une manière telle qu'on limite certaines dépendances; il faut alors s'assurer que les tables et leurs colonnes sont normalisées.

4.3.4 Objets métier

La meilleure manière de représenter les opérations est de créer des procédures sauvegardées ou des fonctions pour chaque opération définie dans le modèle conceptuel. En utilisant cette approche, c'est à l'application d'appeler les bonnes

opérations (Muller, 1999). Il faudra tenir compte que tous les concepts orientés objet ne seront pas implémentés (polymorphisme, avec « overloading » et « overriding ») dans une approche base de données. Il est possible de transformer une opération en une procédure sauvegardée en PL/SQL (Muller, 1999). La méthode (l'implémentation d'une opération en UML) a au moins une entrée (un objet, représentant une ligne du tableau) sur lequel il peut jouer. Cependant, toutes les bases de données ne possèdent pas de procédures sauvegardées. Car elles ne sont pas définies de manière explicite dans la version SQL3. Chaque fournisseur l'implémente à sa manière. Ainsi, dans le cadre du SIGEC, les développements doivent se faire sur des outils n'utilisant aucun langage propriétaire. Toutes les bases de données n'ayant pas de procédures sauvegardées, nous n'avons pas voulu être dépendants d'un langage de programmation qui dépend d'une plate-forme ou d'un outil. Ainsi, les opérations qui se trouvent dans les classes, au lieu de les implémenter dans la base de données *Microsoft Access* comme procédures sauvegardées en VB, elles sont plutôt implémentées au niveau du serveur d'application. Selon la méthode d'implémentation et le langage de programmation utilisé, on n'est pas obligé de recréer la même structure de classe que celle définie dans le modèle conceptuel. On pourrait par exemple créer une seule classe au niveau application qui va regrouper toutes les opérations. Mais, ce n'est pas la manière la plus efficace de procéder en matière d'implémentation. Généralement, dans les applications orientées vers le web, les opérations sont incluses dans une page de scripts aussi appelée *composante*. Cette page script (côté serveur, puisqu'elle doit manipuler les données de la base de données qui se trouve également côté serveur) génère une ou plusieurs pages client. Ainsi, les opérations d'une classe peuvent se trouver dans une ou plusieurs composantes (objet métier) écrites en PHP. La Figure 4.10 montre l'implémentation des opérations dans les différentes pages PHP et ASP.

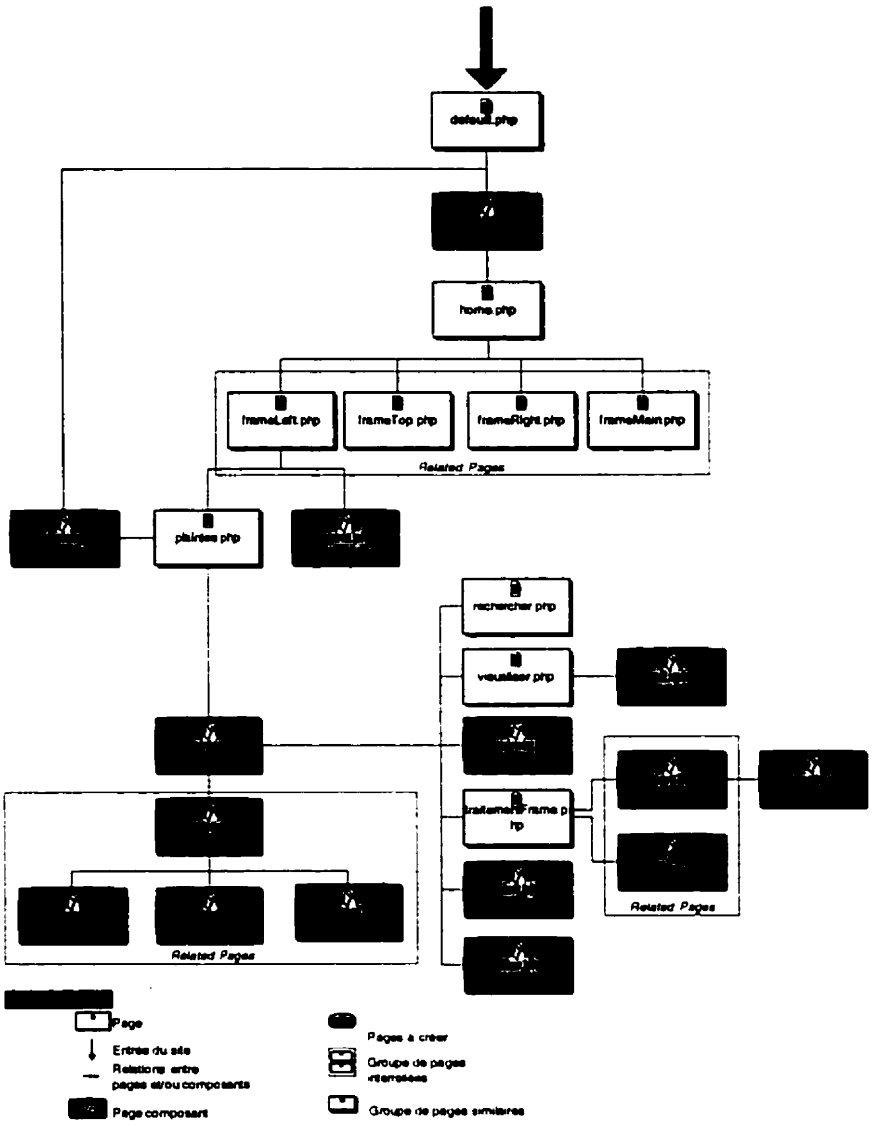


Figure 4.10 Schéma du prototype web pour la gestion des plaintes

Les composantes sont des pages qui implémentent la logique de programmation. Elles effectuent des opérations, présentent une certaine interface aux autres composantes et aux utilisateurs. Les composantes les plus importantes seront décrites dans les lignes qui suivent.

Tout d'abord, la composante d'abstraction aux données permet l'accessibilité à trois différentes bases de données sans souci de la syntaxe de connexion et de manipulation de chaque base de données. Cette composante est très importante car elle fait abstraction des différences entre les bases de données. Elle est essentiellement réalisée grâce aux principes de l'héritage et du polymorphisme. Malheureusement, « l'overloading » et « l'overriding » ne sont pas permis dans le langage PHP, ce qui affecte grandement son utilisation. Avec un langage purement orienté objet tel Java, cela aurait été beaucoup plus facile.

La classe *CDatabase* illustrée à la Figure 4.11 se spécialise en COdbc, CMysql, et CPgsql. Afin qu'il y ait spécialisation avec le langage PHP, toutes les opérations des *classes filles* doivent avoir la même signature que les opérations réciproques de la *classe mère*; il peut y avoir surdéfinition, mais pas de polymorphisme. Cette composante est disponible pour faciliter l'accès aux données.

La composante *CPlaintes.obj* utilise les services de la composante pour l'accès aux données. Elle est définie à la Figure 4.12. La plupart des fonctions de la Figure 4.12 permettent de concevoir l'interface avec l'utilisateur, elles réalisent la page de visualisation des plaintes, de navigation sous forme de table. La fonction *AddTableHead($rs,$nf)* permet d'écrire le nom des colonnes d'une requête donnée quelle qu'elle soit. Elle ajoute sur cette colonne les hyperliens permettant d'ordonner par ordre croissant ou décroissant n'importe quelle colonne d'un tableau. Cette fonction prend en paramètre le résultat d'une requête (un ensemble d'enregistrements) et le nombre de champs que contient le tableau de sortie. Il fait simplement une itération sur les colonnes présentes en prenant soin de masquer les colonnes demandées par la fonction *HideColumn($idColumn)* qui, elle, prend en paramètre le numéro de la colonne à cacher.

```
classCDatabase{
        var$host="";
        var$user="";
        var$password="";
        var$database="";
        var$linkid="";
        var$error="";
        var$errorNo="";

        //Constructeur
        functionCDatabase(){}

        //Ouvre une connexion avec le serveur de BD
        functionSQL_connect($h,$u,$p){}

        //Ouvre une connexion permanente avec le serveur de BD
        functionSQL_pconnect($h,$u,$p){}

        //Sélectionne la BD courant à celle spécifiée
        functionSQL_select_db($db){}

        //execute la requête spécifiée
        functionSQL_query($q){}

        //Libère la mémoire utilisée par le résultat
        functionSQL_free_result($res){}

        //Retourne le nombre de lignes dans le résultat
        functionSQL_num_rows($res){}

        // Copie la ligne suivante dans un tableau
        functionSQL_fetch_array($res){}

        // Ferme la connexion à la BD
        functionSQL_close($link){}

        //Retourne le nombre de lignes affecté par la requête
        functionSQL_affected_rows($res){}

        //Se déplace à l'enregistrement spécifié dans le résultat
        functionSQL_data_seek($res,$row){}

        //Retourne le nombre de champ dans le résultat
        functionSQL_num_fields($res){}

        //Retourne le nom du champ à l'index spécifié
        functionSQL_fieldname($res,$field){}

        //initialise les erreurs
        functionClearError(){}
}
```

Figure 4.11 Classe CDatabase

La fonction *AddTableBody($rs,$nf,$nbR,$offs),* quant à elle, reçoit en paramètres le résultat d'une requête, le nombre de champs de la requête et des indicateurs pour la

navigation : *$nbR* et *$offs*. *$nbR* indique le nombre d'enregistrements à montrer, *$offs* spécifie à partir de quel nombre d'enregistrements il faut commencer à afficher à l'utilisateur.

```
classCPlaintes{
        var$db1="";
        var$sqlQuery="";
        var$linkId="";
        var$resultset="";
        var$htmlResponse="";
        var$numFields="";
        var$title="hui";
        var$numRow="";          // Nombre de lignes retourné par le résultat
        var$nbRow=15;           // Nombre de ligne à montrer
        var$offset=1;           // Position du curseur
        var$scriptName="";      // Nom du  script à appeler pour exécuter une fonctionnalité
        var$endResultset=0;
        var$tmpOrder="";        // Direction du champ à ordonner (ASC ou DESC)
        var$emplColor="";
        var$arrColHide="";      // Pour cacher les colonnes du tableau
        var$showTools="";       // Pour cacher ou montrer les outils d'édition

        function CPlaintes(){}

        functionOpenConnexion(){}

        functionCloseConnexion(){}

        functionHideColumn($idColumn){}

        function View(){}

        function  Write($txt){}

        function AddHtmlHead(){}

        functionAddTableHead($rs,$nf){}

        functionAddTableBody($rs,$nf,$nbR,$offs){}

        function AddHtmlFoot(){}

        function NewLine($i){}

        function AddNavigationBar(){}

        function AddGestionBar(){}

        function AddNew(){}

        function AddLegende(){}
}
```

Figure 4.12 Classe CPlaintes

Malheureusement, comme la version actuelle de PHP n'assure pas la persistance des objets, tout au long d'une session, les attributs d'un objet appartenant à un utilisateur

sont soit sauvegardés dans la base de données (à la prochaine requête, ces attributs seront chargés en mémoire), soit envoyés à l'utilisateur par l'URL des fichiers demandés ou par « cookies ». Dans tous les cas, on est obligé d'assurer la persistance des objets par divers mécanismes de sauvegarde. L'envoi des attributs à l'usager a été adopté. D'autres fonctions servent à créer la barre de navigation et la barre de gestion *function AddNavigationBar()* et *function AddGestionBar()* respectivement. Pour des raisons de développement rapide, les fonctionnalités de saisie, recherche, suppression n'ont pas été réalisées par cette composante mais par d'autres. Cela a tendance à diminuer la complexité de la composante *CPlaintes*, mais en contrepartie augmente le nombre de composantes.

4.3.5 Implémentation de la sécurité

Tout d'abord, le site requiert un point d'entrée unique. Tout utilisateur qui va essayer de rentrer par une autre page va se faire éjecter. Car dans chaque page se trouve un script de vérification des utilisateurs. Ce dernier permet de vérifier qu'un utilisateur est bel et bien un employé, et de quels groupes il fait partie. Si le temps passé par un utilisateur sans interagir avec l'application est supérieur à un certain temps limite, il est prié de s'enregistrer une fois de plus. De cette manière, on peut réellement contrôler les ressources. Si un utilisateur n'a pas accès à une page, il est redirigé vers une autre page. S'il n'a pas accès à une fonctionnalité parce qu'il ne fait pas partie du groupe, l'accès sera refusé. Cela se fait par une recherche dans la base de données, dans la table des groupes et de leurs parents. Si un utilisateur n'est pas présent dans un groupe et également tous les groupes parents, alors c'est un utilisateur invalide pour cette fonctionnalité. La Figure 4.13 montre un script de sécurité inséré dans l'en-tête de chaque page à protéger.

```
if(!empty($validUser)  &&  $validUser=="false"){
        header("Location:http://$HTTP_HOST/Sproto1/logout.php?validUser=false");
        exit();
}

if(!empty($lastTime)){
        $elapsedTime=time()-$lastTime;
        $lastTime=time();
}else{
        header("Location:http://$HTTP_HOST/Sproto1/logout.php?validUser=false");
        exit();
}

if(!empty($lastTime)  &&  $elapsedTime>3600 ){
        header("Location:http://$HTTP_HOST/Sproto1/logout.php?timeElapsed=true");
        exit();
}
```

Figure 4.13 Script de sécurité requis dans chaque page

4.4 Mise en œuvre du prototype

Nous présentons les interfaces personnes-machines qui découlent de l'implémentation du prototype du SIDEX Égout. La Figure 4.14 montre l'interface d'entrée du SIGEC. On accède à l'interface du module de gestion des plaintes à travers cette page d'entrée. Cette centralisation des utilisateurs autour du SIGEC permet de contrôler l'environnement en y apportant un certain niveau de sécurité.

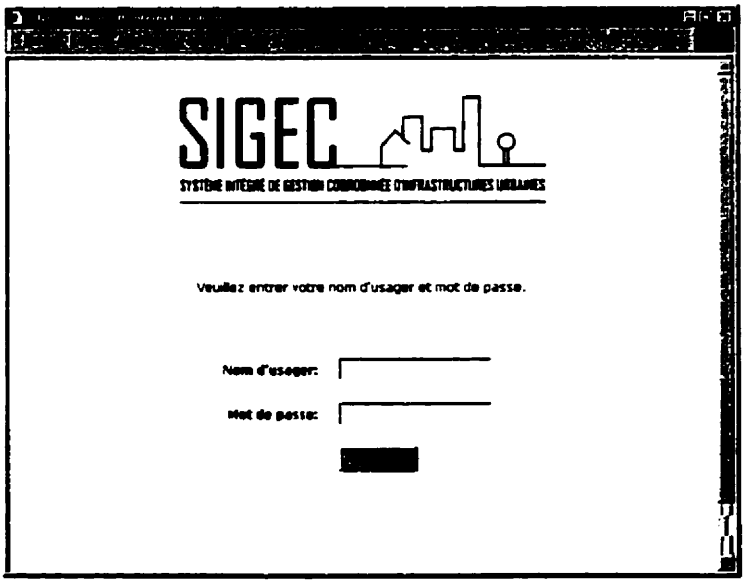


Figure 4.14 Interface d'entrée du SIGEC

La Figure 4.15 montre l'interface principale du module de gestion des plaintes du SIDEX Égout. On y retrouve entre autres un élément de navigation qui permet de visualiser les enregistrements voulus par le gestionnaire des plaintes. Il n'est d'aucune utilité d'afficher des milliers de plaintes à l'écran d'un gestionnaire. Pour faciliter sa tâche, celui-ci peut ordonner chaque champ du tableau de bord. Plusieurs menus se présentent à lui et permettent : la saisie, la suppression, l'édition, la visualisation et la recherche de plaintes. Une aide en ligne ainsi qu'une légende est présente. Cette dernière spécifie les plaintes qui ont été saisies par le gestionnaire en question, les plaintes qui lui sont destinées, ainsi que les plaintes saisies par lui et destinées à lui.

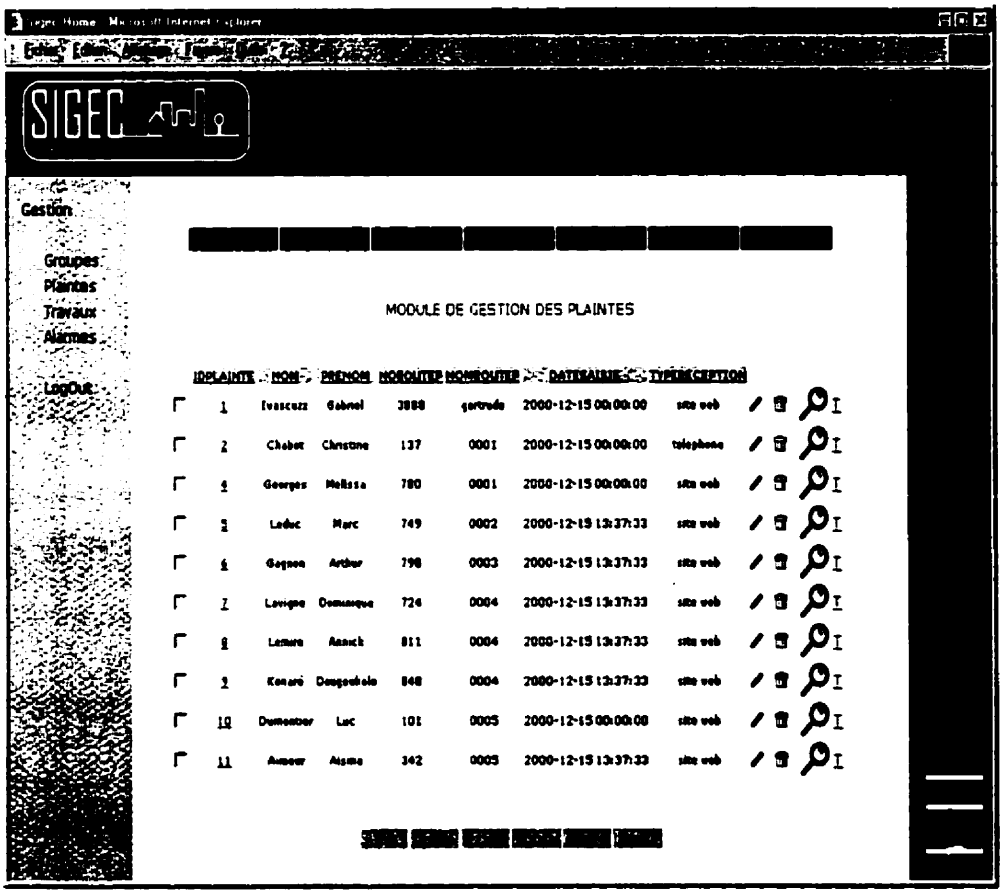


Figure 4.15 Interface principale du module de gestion des plaintes

La Figure 4.16 montre l'interface de saisie d'une plainte. On y voit les éléments essentiels qui ont été définis dans le modèle de données. De plus, l'utilisateur peut apporter une meilleure précision en sélectionnant différents indicateurs au niveau des infrastructures urbaines. Ces indicateurs définissent les règles de dégradation de l'infrastructure en question. Une gestion centralisée au niveau des plaintes est nécessaire.

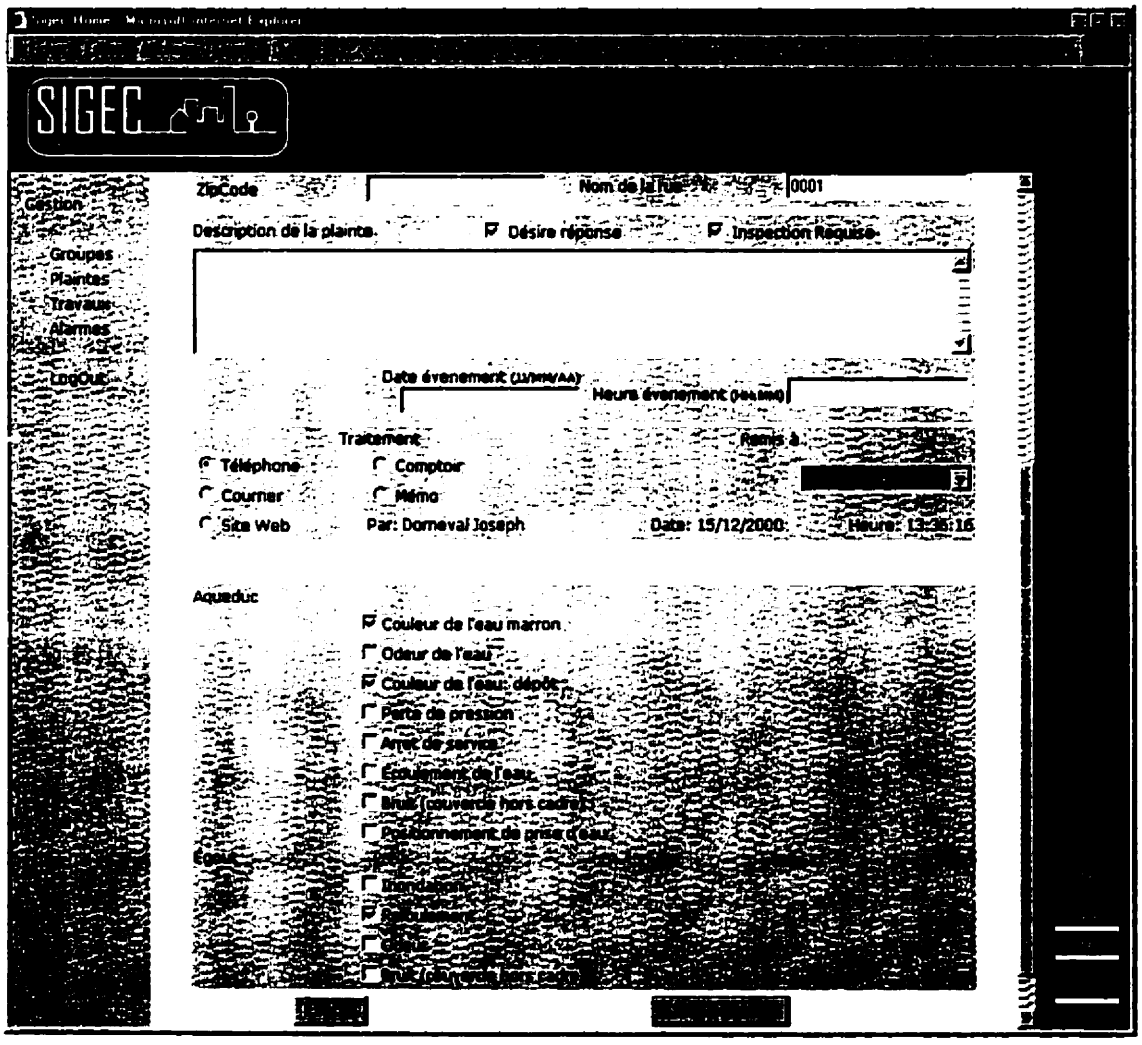


Figure 4.16 Interface de saisie d'une plainte

La Figure 4.17 montre le traitement associé aux plaintes similaires. S'il y a eu une inondation et que cinquante personnes déposent une plainte à ce sujet, dès que la première plainte est résolue, alors toutes les plaintes similaires à cette dernière (i.e. ayant les mêmes caractéristiques) sont considérées comme traitées. Cet écran montre les plaintes similaires dans la partie supérieure avec le degré de similarité. Le choix est laissé au gestionnaire de plaintes de copier ou non le même traitement.

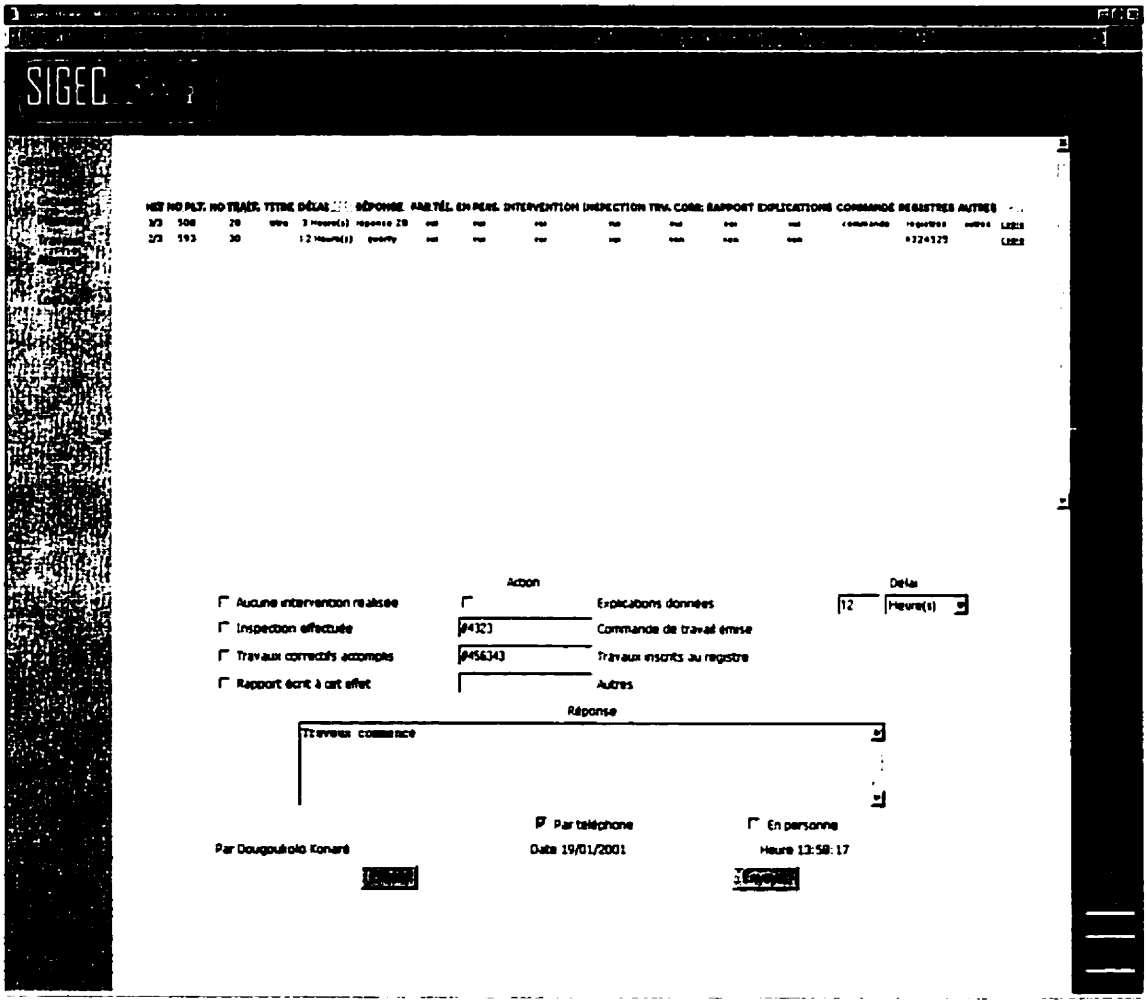


Figure 4.17 Interface de recherche de plaintes similaires

La Figure 4.18 présente le module SIG en action. Grâce à la composante *mapX* de la compagnie *MapInfo*, on arrive à identifier de manière précise n'importe quel élément d'inventaire d'une infrastructure donnée. Dans cette figure, on visualise une requête à référence spatiale. C'est une sélection circulaire autour d'un point. Elle retourne la liste des tronçons de rues présents dans la sélection circulaire.

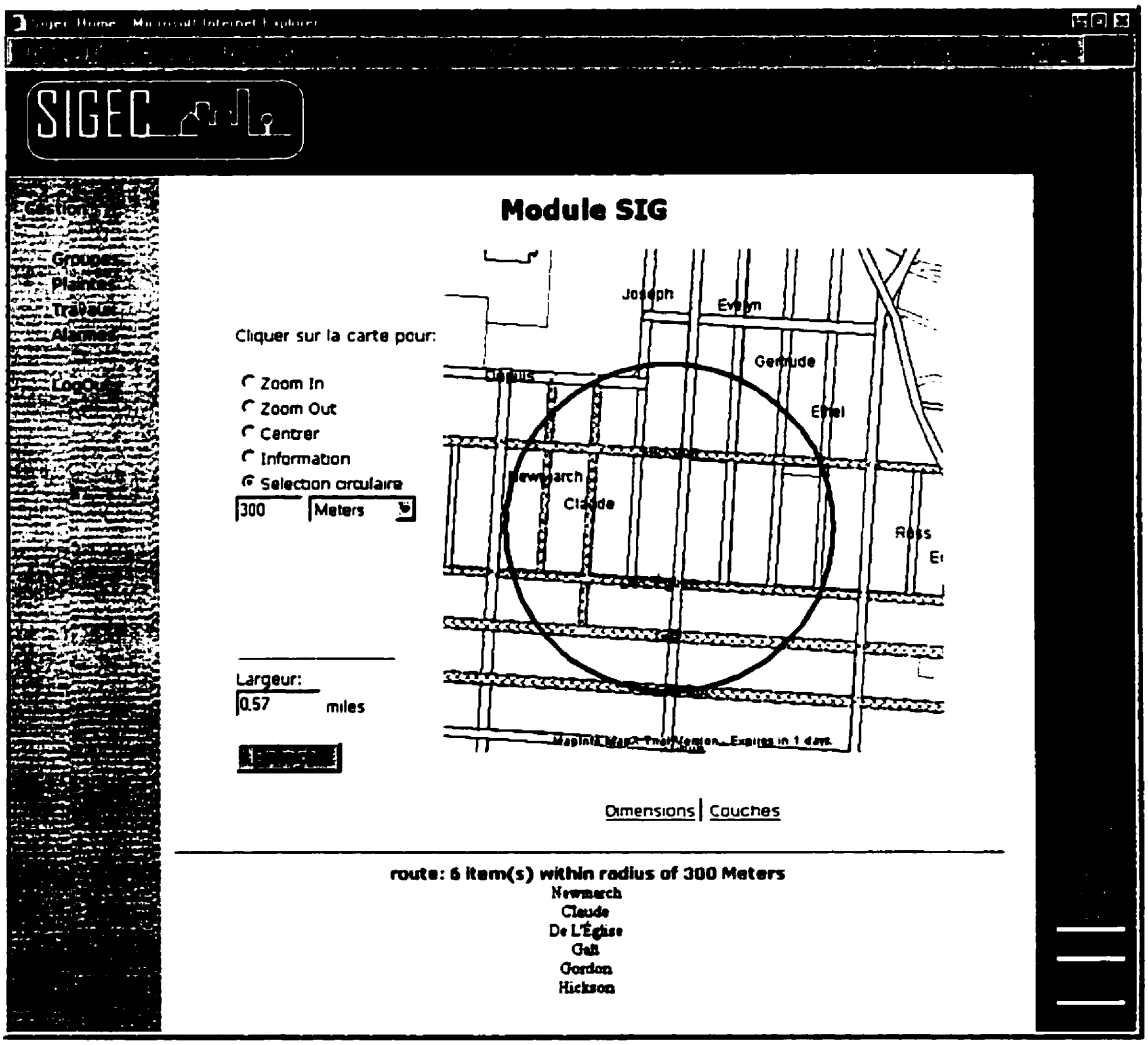


Figure 4.18 Interface de visualisation des éléments d'inventaire

CHAPITRE V
CONCLUSION

Ce chapitre présente une synthèse des travaux effectués dans le cadre de ce mémoire. Une discussion se fera sur les limitations de cette recherche ainsi que les travaux futurs à réaliser.

5.1 Synthèse des travaux

Ce mémoire a présenté les travaux sur la contribution à un système intégré d'exploitation (SIDEX) générique d'infrastructures urbaines avec composantes mobiles multimédia. Nous avons établi une analyse suivant les principes orientés objets du domaine d'application. Une liste d'infrastructures potentielles a été établie et seulement trois d'entre elles ont attiré notre attention pour faire ressortir les éléments intrinsèques aux infrastructures urbaines : le réseau d'égout ou d'assainissement, le réseau d'aqueduc et le réseau de chaussée. Après une analyse approfondie des besoins des municipalités en matière de gestion et d'exploitation des infrastructures urbaines, une liste de fonctionnalités pour chaque système urbain a été établie, fonctionnalités que les SIDEX spécifiques et le SIGEC doivent supporter. Cette étape de spécification formelle du processus de développement de logiciel a permis de définir les contours d'une infrastructure générique ainsi que des fonctionnalités génériques permettant son exploitation. Par la suite, une analyse approfondie en UML selon les principes orientés objets a été effectuée. Cela a permis de définir les principaux acteurs, le modèle statique représenté par le diagramme de classe, ainsi que le modèle dynamique à travers les diagrammes des scénarios. La phase d'analyse a évolué et donné naissance au modèle conceptuel de donnée du SIDEX générique. Cette phase d'évolution du modèle d'analyse vers un modèle conceptuel de donnée a été choisie afin de limiter la

production et la maintenance de plusieurs diagrammes. Le modèle conceptuel également appelé modèle objet, comme son nom l'indique, est plus proche de la machine que de la réalité. C'est une représentation du monde réel dans le monde informatique. Contrairement aux méthodes traditionnelles de développement ou l'on séparait le traitement des données, on a préféré utiliser les principes orientés objets pour regrouper le modèle conceptuel de traitement et de donnée en un seul modèle conceptuel plus proche du monde réel mais décrit en terme informatique, ce qui réduit le temps de développement. Le SIDEX générique est alors défini de manière conceptuelle et sert de cadre de travail (framework). Il est instantié pour chaque SIDEX spécifique. Le processus d'instanciation consiste à rajouter tous les éléments manquants au modèle conceptuel générique pour trouver un modèle conceptuel spécifique. Cette étape d'instanciation est bien sûr précédée par une analyse sommaire du SIDEX spécifique, suivi d'une mise à jour du modèle de conception. Une fois le modèle conceptuel du SIDEX spécifique terminé, on peut passer à l'étape d'implémentation. Un prototype de SIDEX pour l'égout a été implémenté en partie à l'aide des langages de scripts PHP et ASP. Seule la gestion des plaintes du SIDEX Égout est implémentée et permet de valider l'approche de développement de SIDEX par instanciation du SIDEX générique. Tout au long du développement, l'accent a été mis sur un développement modulaire à base de composantes prônant ainsi la réutilisation et l'indépendance au niveau de la conception. La composante DAL permet la connexion à n'importe quelle base de données et l'interrogation des données. Elle sera réutilisable dans d'autres modules par d'autres composantes.

5.2 Limitations des travaux

Bien que suscitant beaucoup d'enthousiasme dans le milieu urbain, cette approche de conception de SIDEX par instanciation du SIDEX générique possède quelques désavantages. En effet, contrairement au système ad hoc conçues pour la gestion des infrastructures urbaines, le SIDEX générique devrait mieux intégrer les objectifs de minimalité, de complétude et de maintenance.

Le SIDEX générique, pour être réellement générique, doit tenir compte de toutes les infrastructures urbaines possibles. Or, jusqu'à maintenant, il ne tient compte que d'un nombre restreint d'infrastructures, ce qui peut affecter le critère de complétude et de minimalité de fonctionnalités. Comment peut-on savoir qu'avec l'ensemble des fonctionnalités, le SIDEX générique répond à l'univers du problème? Comme nous l'avons montré dans les chapitres précédents, le processus de conception du SIDEX générique correspond à une série d'étapes itératives. En effet, l'analyse et la conception du SIDEX générique sont nourries par les tests après implémentation, mais aussi par l'instanciation des différents SIDEX. Ce processus d'instanciation est en fait une phase d'analyse et de conception spécifique au SIDEX. C'est une phase plus courte car elle se base sur l'analyse et la conception du SIDEX générique. Le modèle générique se base sur un ensemble fini et limité de systèmes urbains. La propriété de minimalité est due au fait que les systèmes urbains sélectionnés sont assez représentatifs de tous les systèmes urbains. C'est ainsi que les propriétés intrinsèques aux systèmes urbains ont été définies dans les sections précédentes et servent de modèle de base au SIDEX générique. Trop de fonctionnalités conduiraient à enfreindre le critère de minimalité du SIDEX générique, car si une nouvelle infrastructure urbaine est instanciée, certaines fonctionnalités ne seraient pas applicables.

Quant au critère de complétude, il ne sera atteint en totalité qu'en explorant l'univers de tous les systèmes urbains. Cependant, le but du générique étant d'accélérer la création des SIDEX spécifiques par instanciation, il faut travailler sur un minimum de systèmes urbains qui caractérisent l'ensemble des systèmes urbains. Tout nouveau système urbain sera confronté au générique afin d'améliorer ce dernier. Concevoir un SIDEX générique avec des fonctionnalités propres à tous les systèmes urbains deviendrait trop coûteux lors des étapes d'analyse et de conception, bien que pratiquement cela serait le système idéal, car tous les SIDEX sont instanciés avec seulement les fonctionnalités qui leur sont propres.

L'étape d'implémentation nécessite la conception du schéma relationnel de la base de données. La conception de la structure de données permanente du SIDEX

nécessite un passage du modèle conceptuel en UML à un modèle entité-relation. Cette étape supplémentaire, bien que presque automatique dans certains cas, nécessite un certain temps qui influence le coût de développement des SIDEX par une approche générique. Il est important de bénéficier d'outils CASE facilitant le processus de développement. Il faut constamment faire du « reverse engineering » à chaque évolution d'un modèle. Cette étape très importante permet de maintenir à jour les modèles qui vont être instanciés sans trop de difficulté.

De plus, du fait de la séparation des données et du traitement lors de l'étape d'implémentation, faute de matériel, la recherche ne s'est pas étendue à toutes les plates-formes. L'utilisation du langage PHP est un plus pour le développement des SIDEX. Ces derniers ayant une interface web, le langage PHP est très efficace. Cependant, pour un développement à base de composantes, il est préférable d'utiliser un langage fortement orienté objet tel Java. Avec PHP, certains concepts ne peuvent être appliqués tels le polymorphisme et l'héritage multiple, et freinent considérablement le développement.

5.3 Indication de recherches futures

Les développements effectués dans le cadre de ce mémoire ont porté sur la conception d'un SIDEX générique. Comme preuve de concept, nous avons développé par instanciation du SIDEX générique un prototype de SIDEX appliqué au réseau d'égout ou d'assainissement pour l'exploitation et la gestion des plaintes. Les recherches futures devraient s'orienter vers la consolidation du SIDEX générique. En effet, nous avons remarqué au cours du développement que certains éléments avaient un impact considérable sur l'exploitation des infrastructures urbaines tels les relations avec l'environnement et les sollicitations. Ces concepts du domaine de l'urbanisme méritent d'être pris en compte dans le SIDEX générique ou lors de l'instanciation de chaque SIDEX. L'enrichissement du SIDEX générique passe ainsi par la conception de SIDEX spécifiques à d'autres infrastructures urbaines suivant le processus de conception des applications pour les infrastructures urbaines proposé dans ce mémoire. On peut insister plus particulièrement sur le SIDEX Bâtiment qui, de par sa nature ponctuelle, par

opposition à un système urbain lui-même réseauté, permettra d'enrichir considérablement le SIDEX générique. Le SIDEX générique sera d'autant plus complet qu'il y aura de systèmes qu'il couvre. Afin qu'il n'y ait pas plusieurs versions du SIDEX générique, sa conception et sa maintenance doivent être centralisées. Ainsi dans ce contexte le processus de normalisation se fera par une autorité compétente en la matière. Cette dernière s'occupera alors de maintenir une documentation détaillée des différentes vues du système, de spécifier son mode d'extension, son instanciation.

La conception du SIDEX générique se fait suivant une approche orientée objet avec le langage UML. Toute la richesse de ce dernier repose sur la conception de nouveaux diagrammes afin de présenter de nouvelles vues du SIDEX générique. Bien que les diagrammes enrichissent les différents modèles, leur entretien est une tâche supplémentaire qui peut ralentir considérablement le processus de développement. Il faudra alors maintenir le modèle du SIDEX générique ainsi que les modèles des SIDEX instanciés. Une des approches à ne pas négliger est l'utilisation de bases de données orientées objets (objectivity ou autres). Cette approche permet de maintenir la structure de persistance ainsi que les traitements qui leurs seront attribués. Cela permettra de ne pas concevoir un schéma supplémentaire de conception relationnelle. Dans ce contexte, l'utilisation d'outils CASE apparaît un atout majeur dans le processus de conception. En effet, il permet d'accélérer rapidement le développement ainsi que la maintenance des modèles. L'utilisation du « reverse Engineering » étant monnaie courante dans ce type d'outils aujourd'hui, la maintenance en est simplifiée d'autant.

En ce qui a trait aux méthodes de conception, la réutilisation basée sur l'utilisation de composantes ayant une interface a été employée lors du développement, la communication entre SIDEX n'étant pas encore envisageable. Le système est conçu selon un modèle client/serveur ou requête/réponse. L'utilisateur émet une requête au serveur qui, lui, va traiter la demande et renvoyer la réponse au client à la fin du traitement. Combien de temps prendra le traitement? Cela dépend de la tâche demandée, des performances des équipements (machine et réseau car on est dans un environnement réparti). Lorsqu'un SIDEX particulier a besoin des services d'un autre, il appelle une

méthode d'un autre SIDEX ou va chercher directement l'information dont il a besoin dans la base de données de l'autre SIDEX. Cette technique est fonctionnelle et utilise les mécanismes de communication basés sur les appels de procédures à distance. Tout un environnement est en place pour faciliter cette communication. Une autre approche consisterai à concevoir les SIDEX ainsi que le SIGEC comme un système multi-agent. On recherche à travers les agents l'autonomie qui fait défaut aux systèmes clients/serveurs traditionnels. Ainsi, un utilisateur émettant une requête n'est pas obligé d'attendre indéfiniment une réponse. Un agent est une composante matérielle ou logicielle qui agit sans l'intervention directe d'un humain ou d'une autre entité (Wooldridge, 1994). La mobilité est également un aspect important du système multi-agent dans la gestion des infrastructures urbaines. La disposition géographique des SIDEX peut être améliorée grâce à ce concept. L'intelligence des agents leur permettrait de prendre des décisions en faveur du propriétaire de l'agent.

Finalement, il serait souhaitable de travailler sur différents aspects du prototype tels la sécurité et les interfaces personnes-machines. Tout cela pourrait être complété par une étude de performances des applications urbaines mobiles et multimédias déployées dans l'environnement du SIGEC.

BIBLIOGRAPHIE

AREHART, C., TOSCHI, M., (2000). Professional WAP. Wrox Press Ltd., Birmingham, UK.

BÉGIN, L., (1997). Development and implementation of an integrated infrastructure management system (IIMS). Infra 1997.

BLANPAIN, O., KARNIB, A., (1997). Analyse critique d'un outil d'aide au choix de solutions d'extension ou de restructuration de réseaux d'assainissement. Conférence Internationale sur les nouvelles technologies de l'information pour l'aide à la décision dans le domaine du génie civil, Procedings/Actes/ Vol.2, pp.1075-1082

BOLCER, G., A., (2000). Magi: an architecture for mobile and disconnected workflow. IEEE Internet Computing, vol.4, n°3, pp. 46-54.

BOUZEGHOUB, M., ROCHFELD, A., (2000). OOM La conception objet des systèmes d'information. Hermes, Paris, France.

CASTAGNETTO, J., RAWAT, H., SCHUMANN, S., SCOLLO, C., VELIATH, D., (1999). Professional PHP Programming. Wrox Press, Birmingham, USA.

CERIU, (1999). Rapport d'étape 2, Définition de la structure normalisée Montréal, Canada.

CERIU, (2000). Intégration de la gestion des infrastructures urbaines, guide d'utilisation du cadre de référence. rapport technique, Montréal, Canada.

CONALLEN, J., (2000). Building Web Applications with UML. Addison-Wesley, Harlow, England.

CÔTÉ, B., LEMIEUX, P. F., FORTIER, P., (1997). Développement d'un outil de diagnostic décisionnel pour la gestion des réseaux d'assainissement et de distribution d'eau pour les municipalités de taille moyenne. Univ. De Sherbrooke, Infra 1997, Montréal.

ERLIKH, L., (2000). Leveraging legacy system dollars for e-business. IT Professional technology solutions for the enterprise, vol.2, n°3, pp. 17-23.

ESRI, (16 janvier 2001). ARCFM. HTTP://WWW.ESRI.COM/SOFTWARE/ARCFM/

FAYAD, M., E., HAMU, D., S., BRUGALI, D., (2000). Modeling components and frameworks with UML. Communications of the ACM, vol.43, n°10, pp. 39-46.

FEILER, J., (2000). Application Servers , Powering the Web-Based Enterprise. Morgan Kaufmann.

FINGAR, P., (2000). Component-based frameworks for e-commerce. Communications of the ACM, vol.43, n°10, pp. 61-66.

FONTAINE, A.B., REICH, G.P., ZAIM, V., (1995). Modélisation et conception orientées objet. Masson, Paris, France

GAIN ECONFIDENCE, (1999). The e-business reliability survival guide. Segue Software.

GODIN, R., (2000). Systèmes de gestion de bases de données. Notes de Cours UQAM, Montréal.

HEGARET, P., L., WOOD, L., ROBIE, J., (13 novembre 2000). Texcel (for DOM Level1).http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-00001113/introduction.html

HEYWOOD, I., CORNELIUS, S., CARVER, S., (1998). An Introduction to Geographical Information Systems. Addison Wesley, New York.

JOHNSON, R., A., (2000). The ups and downs of object-oriented systems development. Communications of the ACM, vol.43, n°10, pp. 69-73.

KETTANI, N., MIGNET, D., PARE, P., ROSENTHAL-SABROUXC., (1998). De Merise à UML, Eyrolles. Paris, France.

KOBRYN, C., (2000). Modeling components and frameworks with UML. Communications of the ACM, vol.43, n°10, pp. 31-38.

KOUSHIK, S., JOODI, P., (2000). E-business architecture design issues. IT Professional technology solutions for the enterprise, vol.2, n°3, pp. 38-43.

LAI, M., (1999). Penser objet avec UML et Java. InterEditions, Paris, France.

LARSEN, G., (2000). Component-Based Enterprise Frameworks. Communications of the ACM, vol.43, n°10, pp. 25-30.

LETOURMY, L., PAPIERNIK, T., HÉLAÏLI, A., MARTZEL, X., (2000). Construire une application WAP. Eyrolles, Paris, France.

LIAO, T., (2000). Global information broadcast: an architecture for internet push channels. IEEE Internet Computing, vol.4, n°3, pp. 46-54.

MAPINFO CORPORATION, (1998). MapInfo Professional User's Guide. MapInfo Corporation, New York, USA.

MOTTIER V., (1999). DWSL : un composant logiciel pour la construction de systèmes d'information pour la gestion de l'assainissement urbain. Rapport technique, École polytechnique fédérale de Lausanne, Suisse.

MULLER, R., J., (1999). Database design for smarties, using UML for data modeling. Morgan Kaufmann, San Francisco.

OESTEREICH, B., (1999). Developing software with UML Object-Oriented, Analysis and Design in Practice. Addison-Wesley, Harlow, England.

PEACOCK, R., (2000). Distributed architecture technologies. IT Professional technology solutions for the enterprise, vol.2, n°3, pp. 58-60.

PRESSMAN, R. S., (1997). Software Engineering a practitioner's approach. McGraw-Hill, New York, USA.

SPARLING, M., (2000). Lessons learned through six years of component-based development. Communications of the ACM, vol.43, n°10, pp. 47-53.

VSA ASSOCIATION SUISSE DES PROFESSIONNELS DE L'EVACUATION DES EAUX, (1999). VSA-DSS. Zürich, Suisse.

ANNEXE A
LISTE DES FONCTIONNALITES DETAILLEES

$O_{1,1}$ – Identification d'un tronçon de rue

Référencer les données par tronçon de rue et codification polygonale des intersections à partir des informations à références spatiales disponibles et relatives à ce tronçon.

Intégration de fonctionnalités d'identification GIS (à expliciter)

O1,2 – Saisie d'information à références spatiales

Permettre à l'usager et aux divers équipements, sur le territoire propre au système, l'entrée automatique et manuelle des informations à références spatiales qui sont produites ou disponibles et relatives aux tronçons.

$O_{1,3}$ – Requête d'information à références spatiales

Permettre à l'usager et aux divers équipements, sur le territoire propre au système, l'extraction automatique et manuelle des informations à références spatiales qui sont produites ou disponibles et relatives aux tronçons.

$O_{1,4}$ – Tri

Permettre le tri par éléments fonctionnels et génération de tables relationnelles.

$O_{1,5}$ – Génération de la table de référence

$O_{1,6}$ – Génération et accès à une banque d'images vidéo

$O_{1,7}$ – Génération de formulaires automatiques complétés des télémesures

$O_{1,8}$ – Validation des télémesures dans le réseau

Contrôle d'intégrité des mesures

$O_{1,9}$ – Génération de la table des besoins et de la programmation des travaux

Besoins identifiés

Priorités proposées

Scénarios d'interventions proposées

Coûts/bénéfices des scénarios proposés

Scénarios retenus

Interventions

$O_{1,10}$ – Interaction avec le système de gestion en temps réel

Génération de la table des évènements et des alarmes

Changement de points de consignes

Analyse des tendances

Consultation de l'historique

Gestion des plaintes

Information sur le code de programmation, la date des travaux, les interventions sur le réseau, l'évolution des travaux (programmation, exécution, fin)

$O_{1,11}$ – Génération des interventions et accès aux interventions prévus par des tiers sur des systèmes réseaux adjacents non directement gérés par la ville (NDGV)

$O_{1,12}$ – Mise en correspondance de plan directeur et de plan orthophotonumérique

À partir des relevés terrains et GPS, valider l'intégrité des informations.

Production, perfectionnement et mise à jour du plan directeur sur plan orthophotonumérique

ANNEXE B

CAS D'UTILISATION

Modifier l'état d'une plainte

Objectif : Ce cas d'utilisation montre comment le gestionnaire des plaintes modifie l'état d'une plainte lorsqu'il reçoit de nouvelles informations la concernant

Acteur Primaire : Gestionnaire des plaintes

Acteur Secondaire : BD de plaintes

Condition d'initialisation : La plainte existe dans le système

Condition de terminaison succès : Les champs correspondant du formulaire de plainte sont modifiés et une mise à jour de l'état est effectuée

Condition de terminaison insuccès : La plainte n'est pas modifiée

Scénario de transaction principal

1. Le cas d'utilisation débute lorsque le gestionnaire des plaintes reçoit de nouvelles informations concernant une plainte
2. Le gestionnaire des plaintes recherche la plainte
3. Le gestionnaire sélectionne la plainte
4. Le système affiche le formulaire de plainte
5. Le gestionnaire modifie les champs correspondants du formulaire de plainte
6. Le gestionnaire sélectionne *enregistrer*
7. Le système affiche un message de confirmation pour la modification de la plainte
8. Le gestionnaire de plainte sélectionne *continuer*
9. Le système vérifie la nouvelle information et sauvegarde la plainte dans la base de données. Le cas d'utilisation se termine.

Extension

5.1 Le système imprime la plainte

Exception

Le gestionnaire des plaintes peut annuler la transaction en trois moments :

a) À tout moment avant que le formulaire de plainte ne soit affiché à l'écran, il utilise une nouvelle fonction du système.
b) À tout moment avant de sélectionner *enregistrer* il annule la transaction en sélectionnant *annuler*.
c) Lorsque le message de confirmation est affiché, il sélectionne *annuler*.

Les deux dernières méthodes d'annulation de transaction amènent le gestionnaire de plainte vers la page de travail principale tandis que la première méthode le dirige vers la fonction qu'il a sélectionné. La modification de la plainte n'a pas lieu et le cas d'utilisation se termine.

Commentaire

La seule personne habileté à modifier une plainte est le gestionnaire des plaintes. La demande pour modifier la plainte peut venir du gestionnaire lui-même ou d'un intervenant municipal en charge de l'évaluation de la plainte.

La vérification des informations, lorsque le gestionnaire des plainte sélectionne *enregistrer*, se fait en deux temps. Tout d'abord, le système vérifie que les champs nécessaires à la création de la plainte sont remplis. Ensuite, le système vérifie que les informations entrée sont cohérentes en terme de type de donnée; c'est-à-dire que l'information entrée correspond au type que le champ est apte à recevoir.

Créer une plainte

Objectif : Ce cas d'utilisation montre comment le gestionnaire des plaintes entre une nouvelle plainte dans le système.

Acteur Primaire : Gestionnaire des plaintes

Acteur Secondaire : BD de plaintes

Condition d'initialisation : Une plainte est émise

Condition de terminaison succès : La plainte est entrée dans le système et une mise à jour est effectuée..

Condition de terminaison insuccès : La plainte n'est pas entrée dans le système.

Scénario de transaction principal

Le cas d'utilisation débute lorsque le gestionnaire des plaintes reçoit une nouvelle plainte à entrer dans le système
Le système affiche le formulaire de création de plainte
Le gestionnaire complète le formulaire de plainte avec les informations en main.
Le gestionnaire sélectionne *enregistrer*
Le système affiche un message de confirmation pour la création de la plainte
Le gestionnaire de plainte sélectionne *continuer*
Le système vérifie l'information entrée et sauvegarde la nouvelle plainte dans la base de données
Le cas d'utilisation se termine.

Extension

3.1 Le système imprime la plainte

Exception

Le gestionnaire des plaintes peut annuler la transaction en deux moments :

a) À tout moment avant de sélectionner *enregistrer* il annule la transaction en sélectionnant *annuler.*
b) Lorsque le message de confirmation est affiché, il sélectionne *annuler.*

Les deux méthodes d'annulation de transaction dirigent le gestionnaire de plainte vers la page de travail principale. La plainte n'est pas entrée dans le système et le cas d'utilisation se termine.

Commentaire

Dès qu'une plainte est émise par un plaignant celle-ci est systématiquement entrée dans le système.

La vérification des informations, lorsque le gestionnaire des plainte sélectionne *enregistrer*, se fait en deux temps. Tout d'abord, le système vérifie que les champs nécessaires à la création de la plainte sont remplis. Ensuite, le système vérifie que les informations entrée sont cohérentes en terme de type de donnée; c'est-à-dire que l'information entrée correspond au type que le champ est apte à recevoir.

Supprimer une plainte

Objectif : Ce cas d'utilisation montre comment le gestionnaire des plaintes supprime une plainte du système

Acteur Primaire : Gestionnaire des plaintes

Acteur Secondaire : BD de plaintes

Condition d'initialisation : La plainte existe dans le système.

Condition de terminaison succès : La plainte est supprimée du système et une mise à jour est effectuée..

Condition de terminaison insuccès : La plainte n'est pas supprimée du système.

Scénario de transaction principal

1. Le cas d'utilisation débute lorsque le gestionnaire des plaintes reçoit une demande de suppression de plainte
2. Le gestionnaire des plaintes recherche la plainte
3. Le gestionnaire sélectionne la plainte
4. Le système affiche le formulaire de plainte
5. Le gestionnaire sélectionne *supprimer*
6. Le système affiche un message de confirmation pour la suppression de la plainte
7. Le gestionnaire de plainte sélectionne *continuer*
8. Le système effectue une mise à jour de la base de données et le cas d'utilisation se termine

Extension

Aucune

Exception

Le gestionnaire des plaintes peut annuler la transaction en deux moments :

a) À tout moment avant de sélectionner *enregistrer* il annule la transaction en sélectionnant *annuler.*
b) Lorsque le message de confirmation est affiché, il sélectionne *annuler.*

Les deux méthodes d'annulation de transaction dirigent le gestionnaire de plainte vers la page de travail principale. La plainte n'est pas supprimée du système et le cas d'utilisation se termine.

Commentaire

La demande de suppression de la plainte vient de l'intervenant municipal en charge de l'évaluation de la plainte. Le gestionnaire des plaintes ne fait qu'exécuter sa demande.

Rechercher une plainte

Objectif : Ce cas d'utilisation montre comment un gestionnaire de plaintes recherche une plainte dans le système

Acteur Primaire : Gestionnaire des plaintes

Acteur Secondaire : BD de plaintes

Condition d'initialisation : La plainte existe dans le système.

Condition de terminaison succès : La plainte recherchée est affichée.

Condition de terminaison insuccès : La plainte recherchée n'est pas trouvée.

Scénario de transaction principal

1. Le cas d'utilisation débute lorsque le formulaire de recherche de plaintes est affiché à l'écran
2. Le gestionnaire des plaintes complète le formulaire de recherche selon les informations d'identification qu'il possède sur la plainte
3. Le gestionnaire sélectionne *rechercher*
4. Le système vérifie les informations de recherche entrées
5. Le système envoie une requête d'information à la base de donnée de plaintes
6. Les résultats de la recherche sont affichés à l'écran par le système et le cas d'utilisation se termine.

Extension

Aucune

Exception

Trois types d'exception modifiant le déroulement normal du cas d'utilisation peuvent survenir :

a) Les informations entrées dans le formulaire de recherche ne sont pas cohérentes avec le type des champs correspondant. Le système affiche un message d'erreur. Le gestionnaire corrige les informations entrées et le cas d'utilisation se poursuit
b) Le gestionnaire des plaintes choisit une autre fonction du système et le cas d'utilisation se termine
c) La recherche ne donne aucun résultat et le cas d'utilisation se termine

Commentaire

Il existe une variété de paramètres permettant d'identifier une plainte. Le choix du ou des paramètres à utiliser est déterminé par les informations que possède le gestionnaire des plaintes sur la plainte.

Consulter une plainte

Objectif : Ce cas d'utilisation montre comment un gestionnaire de plaintes consulte une plainte dans le système

Acteur Primaire : Gestionnaire des plaintes

Acteur Secondaire : BD de plaintes

Condition d'initialisation : La plainte existe dans le système.

Condition de terminaison succès : La plainte est affichée.

Condition de terminaison insuccès : La plainte n'est pas affichée.

Scénario de transaction principal

1. Le cas d'utilisation débute lorsque les résultats d'une recherche de plaintes sont affichés à l'écran
2. Le gestionnaire des plaintes sélectionne une plainte
3. Le système affiche la plainte à l'écran et le cas d'utilisation se termine

Extension

3.1 Le système imprime la plainte

Exception

Il peut survenir une erreur au niveau du système, il en résulte que le formulaire de plainte n'est pas affiché à l'écran. Dans cette circonstance, le cas d'utilisation se termine.

Commentaire

Aucun

Imprimer une plainte

Objectif : Ce cas d'utilisation montre comment le gestionnaire de plainte imprime un formulaire de plainte.

Acteur Primaire : Gestionnaire des plaintes

Acteur Secondaire : BD de plaintes

Condition d'initialisation : Le formulaire de plainte est affiché à l'écran

Condition de terminaison succès : Les informations sur la plainte sont imprimées.

Condition de terminaison insuccès : La plainte n'a put être imprimée.

Scénario de transaction principal

1. Le cas d'utilisation débute lorsque le formulaire de plainte est affiché à l'écran
2. Le gestionnaire de plainte sélectionne *imprimer*
3. Le système imprime la plainte et le cas d'utilisation se termine

Extension

Aucune

Exception

Il peut survenir une erreur au niveau du système, il en résulte que le formulaire de plainte n'est pas imprimé. Dans cette circonstance, le cas d'utilisation se termine

Commentaire

Il est possible de penser que dans une version ultérieure du SEG, plusieurs formats d'impressions pour une plainte puissent exister.

Émettre une plainte

Objectif : Ce cas d'utilisation montre comment un plaignant émet une plainte sur le système urbain égout. Une réclamation, qui est un cas particulier où un montant forfaitaire est réclamé, peut découler de cette plainte.

Acteur Primaire : Gestionnaire des plaintes , plaignant

Acteur Secondaire : Aucun

Condition d'initialisation : Un problème existe au niveau du système urbain égout.

Condition de terminaison succès : La plainte à été émise.

Condition de terminaison insuccès : La plainte n'a pas été émise.

Scénario de transaction principal

1. Le cas d'utilisation débute lorsque le plaignant contacte l'intervenant de la municipalité dont le rôle est de recevoir les plaintes
2. Le plaignant énonce l'objet de sa plainte et donne toutes les informations nécessaires.
3. Le gestionnaire des plaintes complète le formulaire de plainte selon les informations en main et le cas d'utilisation se termine

Extension

Aucune

Exception

Deux types d'exceptions peuvent survenir lors de l'émission de la plainte :

a) Le plaintif ne peut joindre l'intervenant responsable de recevoir les plaintes et le cas d'utilisation se termine.
b) Le plaintif ne possède les informations minimales pour compléter la création de la plainte. La plainte n'est pas entrée dans le système et le cas d'utilisation se termine.

Commentaire

Le plaignant peut émettre sa plainte de plusieurs manières :

a) Verbalement, en présence du gestionnaire de plainte
b) Par téléphone
c) Par lettre officielle
d) Par courrier électronique
e) Par le mécanisme d'émission de plainte offert par le SEG

Le gestionnaire des plaintes peut créer directement la plainte dans le système lorsque celle-ci est émis. Il peut arriver des situations où celui-ci doit tout d'abord noter la plainte sur un formulaire de plainte avant de l'entrer dans le système.

Dans le cas où le plaignant émet une réclamation, des montants forfaitaires sont associés à la plainte et son statut est modifié. Autrement dit, une réclamation est traitée différemment d'une simple plainte.

Effectuer requête d'informations plaintes

Objectif : Ce cas d'utilisation montre comment une requête d'informations sur les plaintes est traitée par le système. Son objectif est de permettre l'utilisation des données de la base de données de plaintes selon les besoins du système

Acteur Primaire : Gestionnaire des plaintes

Acteur Secondaire : BD de plaintes

Condition d'initialisation : Les données existent dans la base de données

Condition de terminaison succès : Les informations ont été trouvées et fournies à le gestionnaire des plaintes

Condition de terminaison insuccès : Les informations n'ont pas été trouvées

Scénario de transaction principal

1. Le cas d'utilisation débute lorsque le gestionnaire des plaintes effectue une opération nécessitant une requête à la base de données
2. Le système se connecte à la base de données de plainte
3. Le système effectue la requête à la base de données
4. La base de données traite la requête et retourne le résultats
5. Le système ferme la connexion à la base de donnée et le cas d'utilisation se termine

Extension

Aucune

Exception

Il existe quatre types d'exceptions appliquées à ce cas d'utilisation :

a) La connexion à la base de données de plaintes n'a pu être effectuée et le cas d'utilisation se termine
b) La requête n'a pu avoir lieu, la connexion avec la base de données de plaintes est fermée et le cas d'utilisation se termine.
c) Les résultats de la requête n'ont pu être retournés au système, la connexion avec la base de données de plaintes est fermée et le cas d'utilisation se termine.
d) La connexion avec la base de données de plaintes n'a pu être fermée et le cas d'utilisation se termine.

Commentaire

Une requête à la base de données de plaintes peut prendre plusieurs formes :

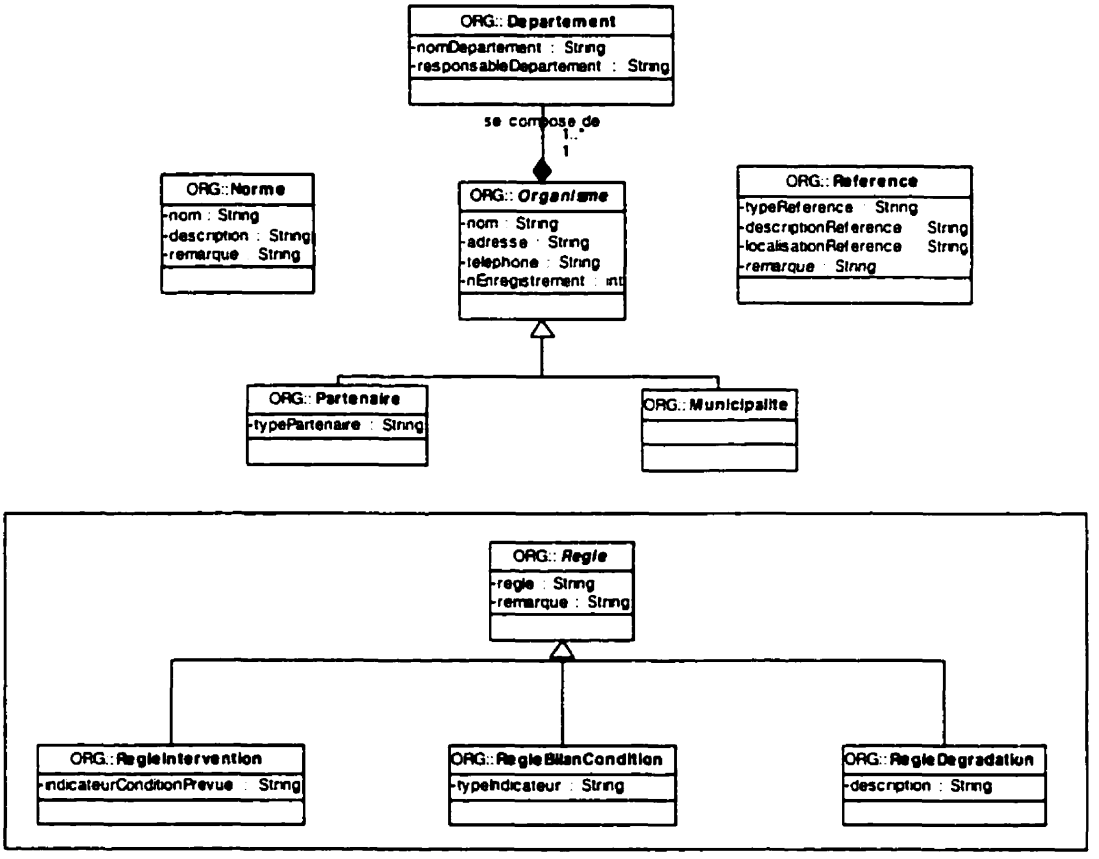
a) Insertion
b) Modification
c) Sélection
d) Supression
e) Etc.

Étant donné l'importante variété de requêtes possibles, il est difficile de toutes les énumérer ici. Toutefois, la requête d'informations est un cas d'utilisation générique pouvant s'adapter à tous ces types de requêtes

ANNEXE C

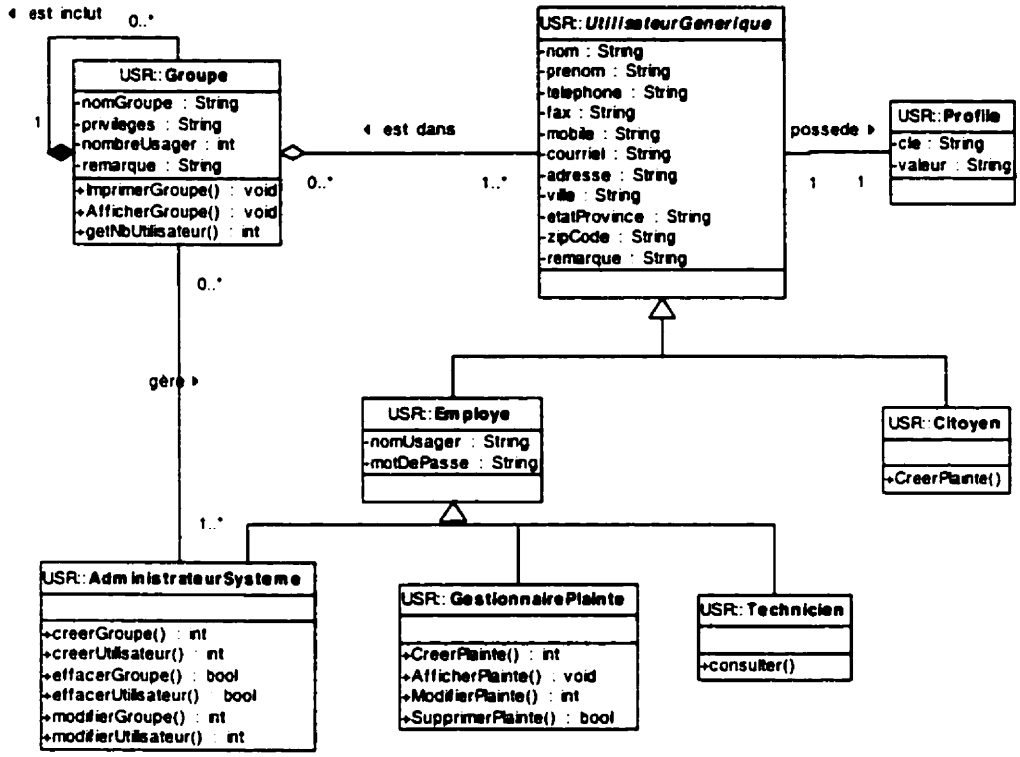
SIDEX ÉGOUT

Package ORGANISATION

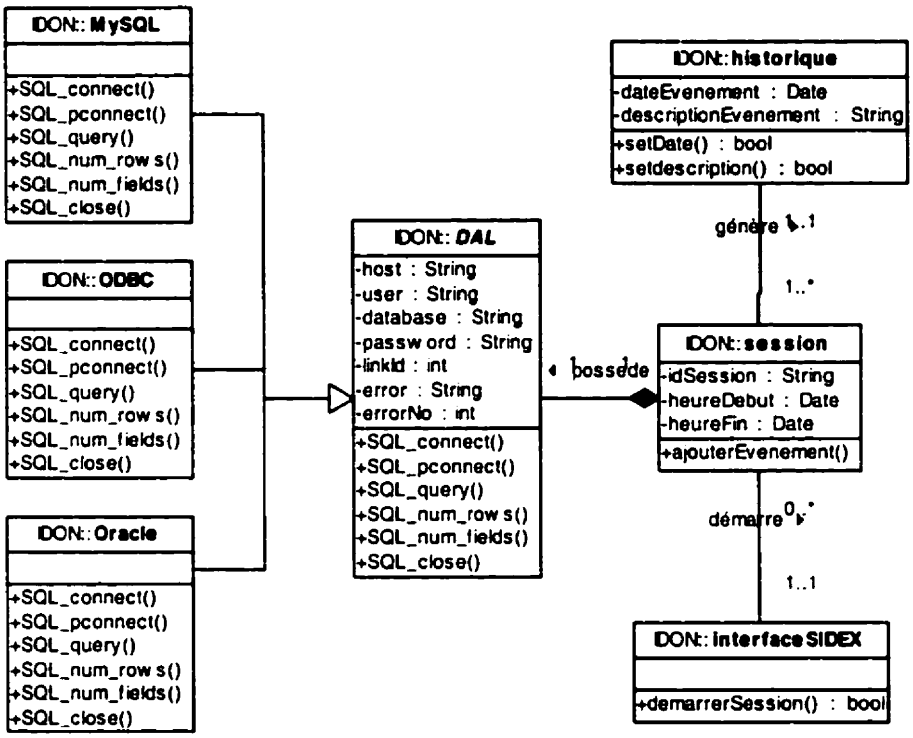


SIAD

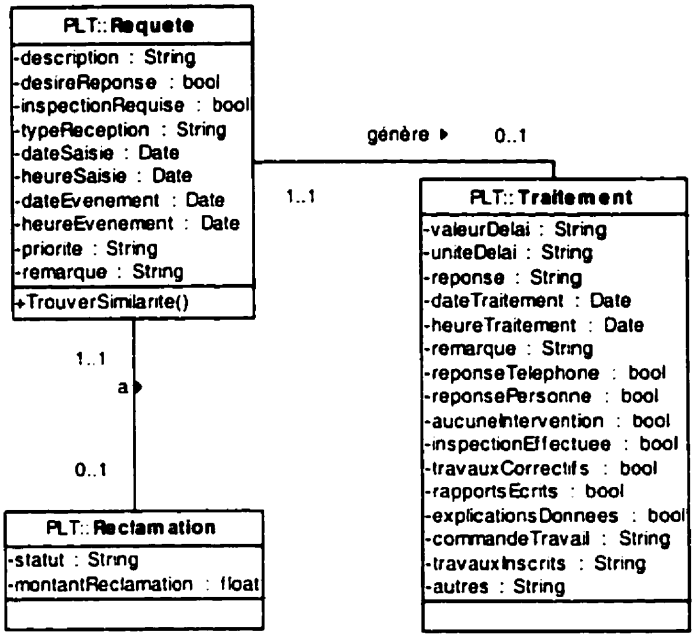
Package USAGER



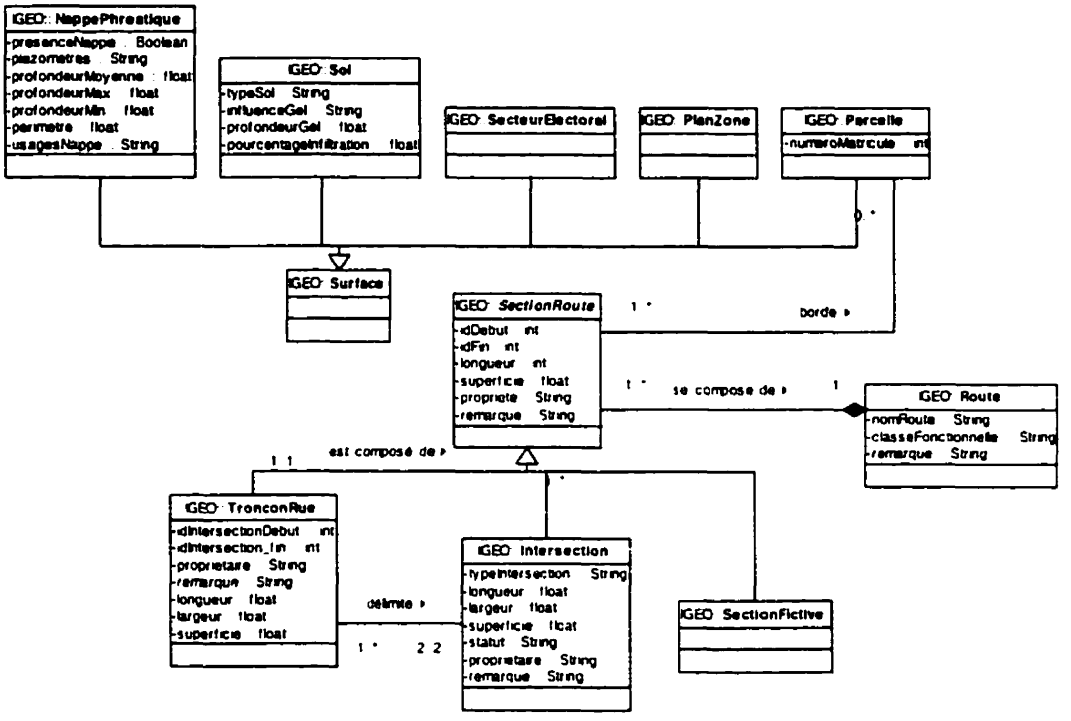
Package INTERFACE_DONNEES



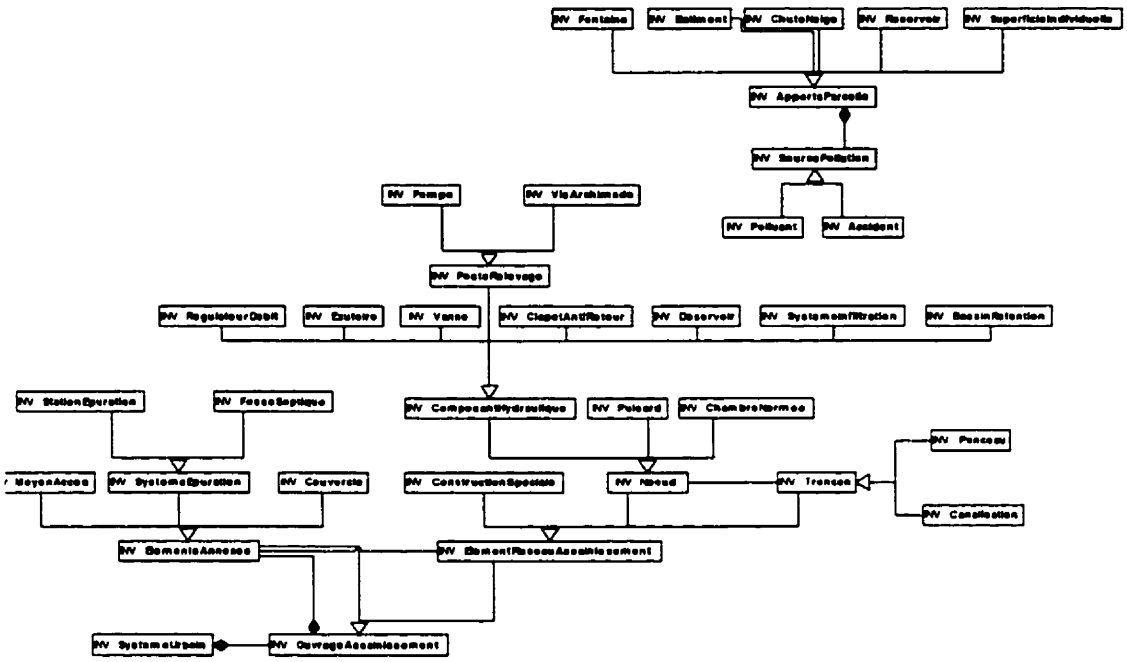
Package PLAINTE



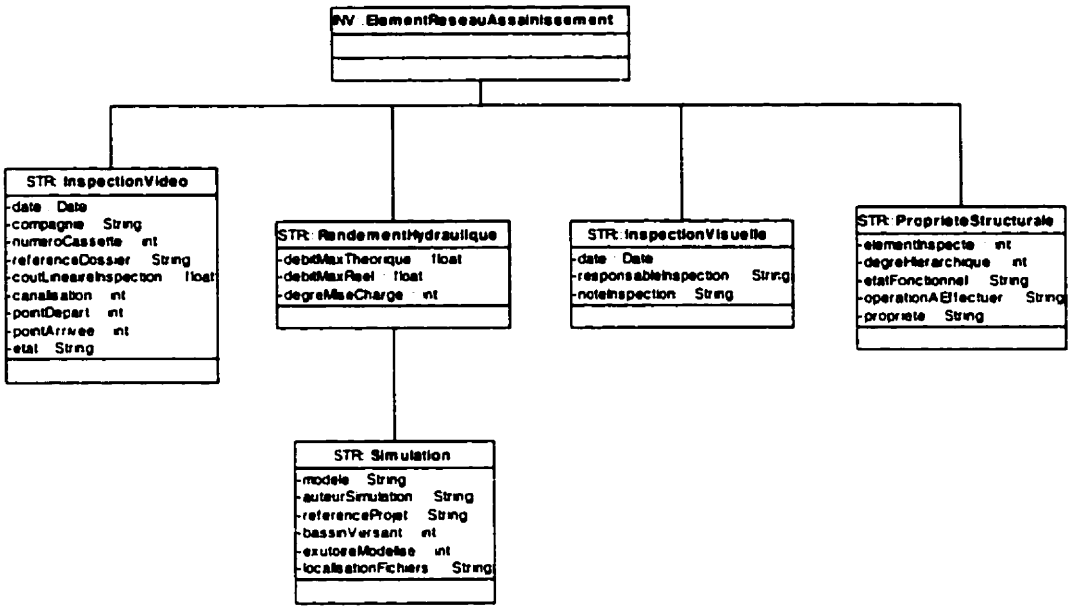
Package IDENTIFICATION_GEOGRAPHIQUE



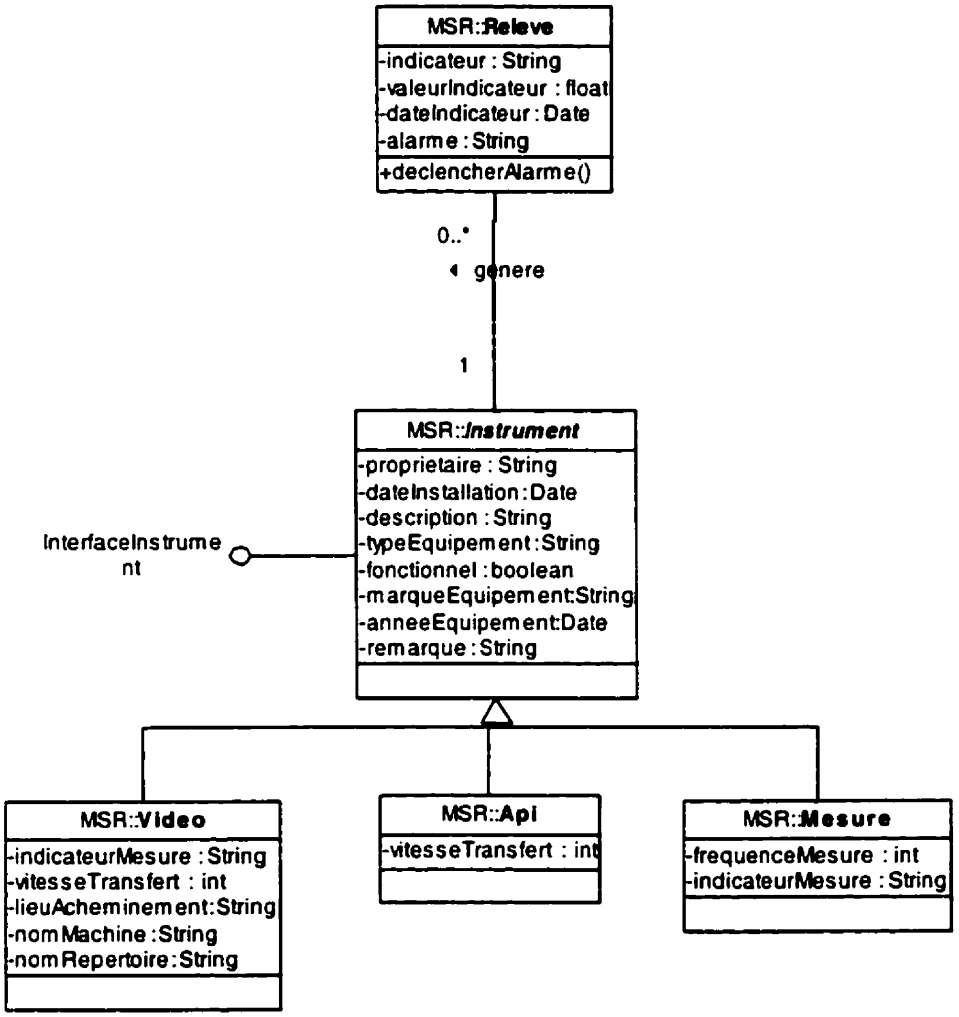
Package INVENTAIRE



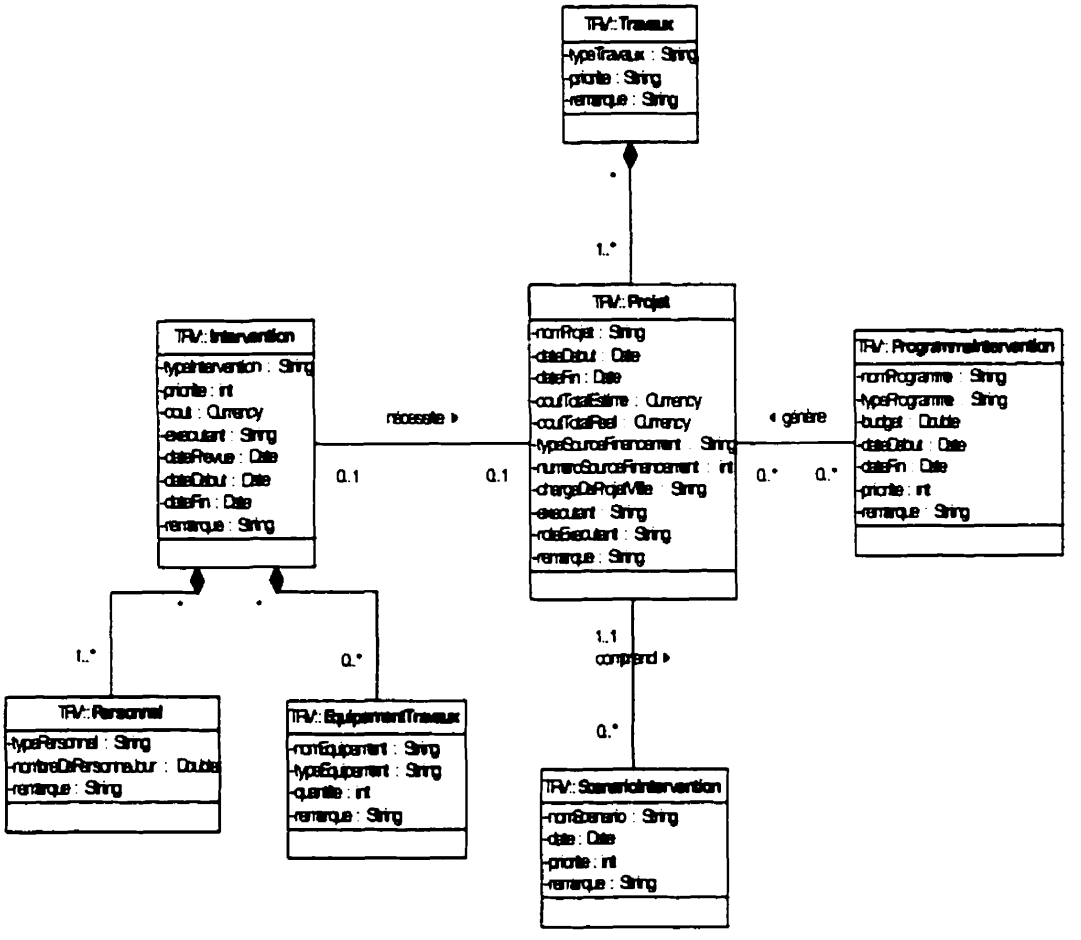
Package PROPRIETE_STRUCTURALE



Package MESURES



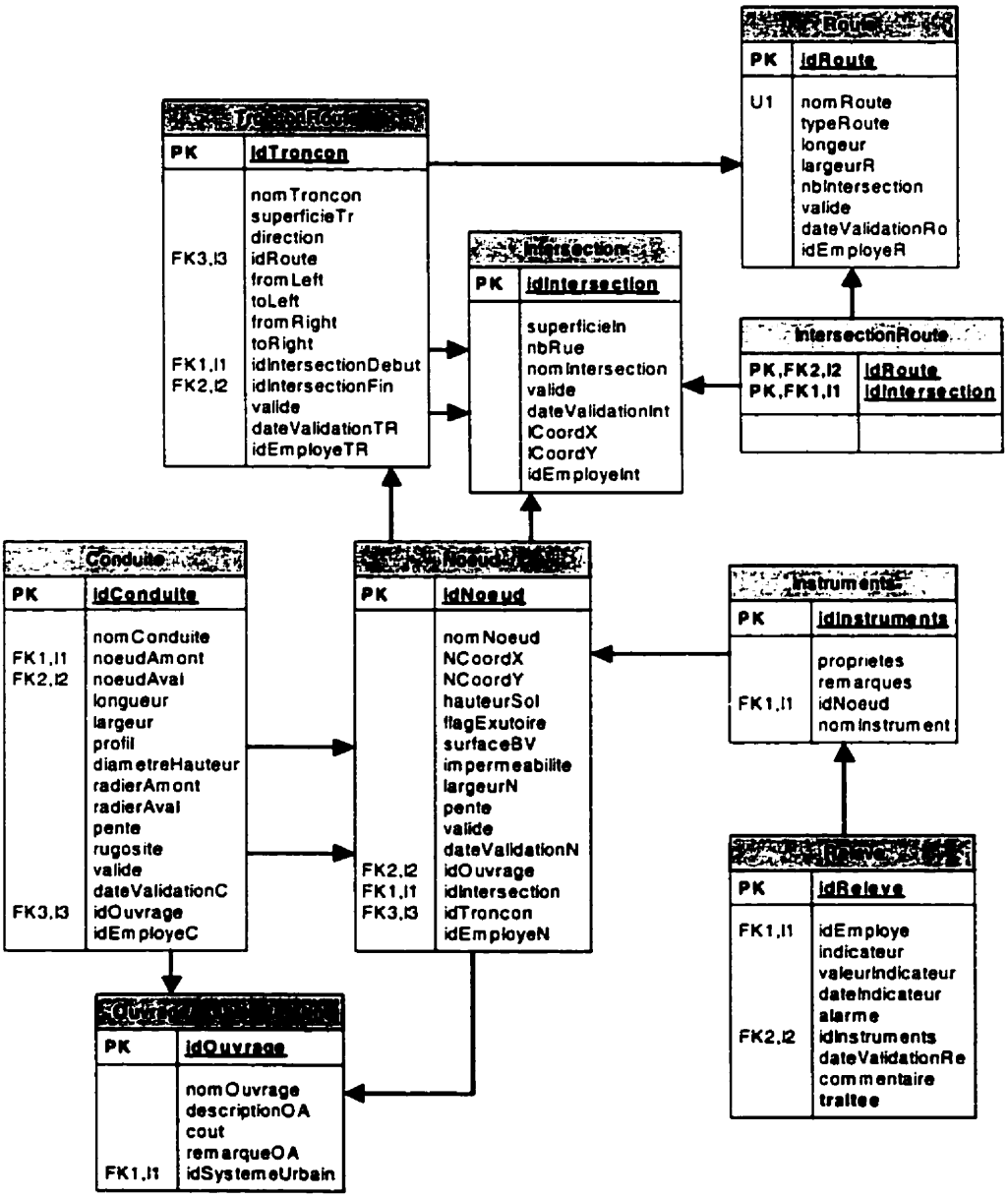
Package TRAVAUX



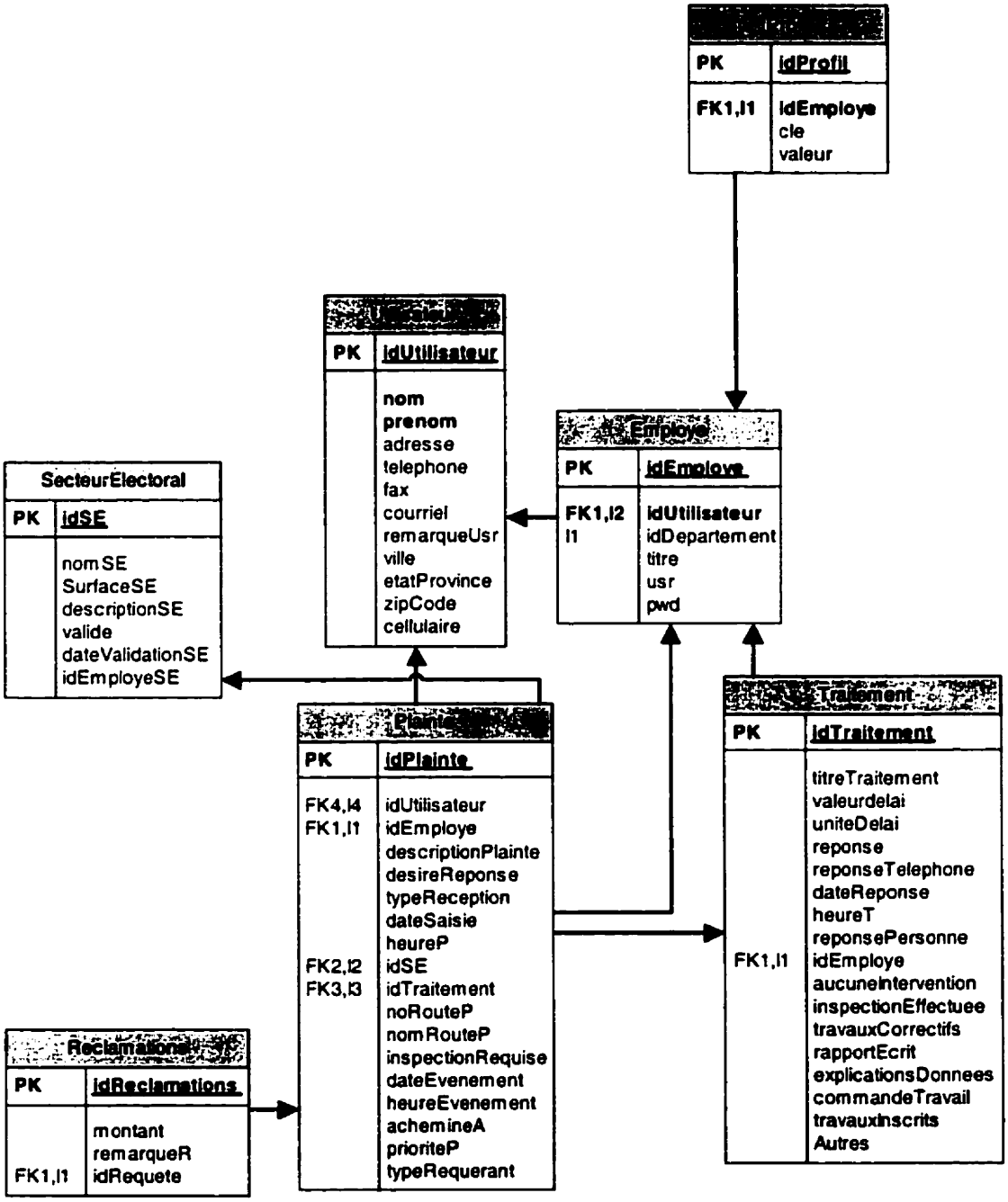
ANNEXE D

SCHEMA DE LA BASE DE DONNEES DU SIDEX ÉGOUT

IDENTIFICATION_GEOGRAPHIQUE ET INVENTAIRE



UTILISATEUR et PLAINTE



ORGANISATION