Du développement à la mise en production d'une application web : le projet MORSE (Module d'Organisation des Ressources Spécialisées et des Équipages)

Patricia Havan

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL ASSOCIE DE NOUMEA

_____

MEMOIRE

présenté en vue d'obtenir

le DIPLOME d'INGENIEUR CNAM

SPECIALITE : Informatique

OPTION : Ingénierie des Systèmes d'Information (ISI)

par

Patricia HAVAN

_____

Du développement à la mise en production d'une application web : le projet MORSE (Module d'Organisation des Ressources Spécialisées et des Equipages)

Soutenu le 14 février 2011

_____

JURY

PRESIDENT : M. le Professeur Jacky AKOKA (titulaire de la chaire informatique du CNAM)

MEMBRES :

M. Henri LAUGIE (auteur, enseignant au CNAM de Nouméa, à l'UNC et au BTS)

M. Jacques THILLIER (enseignant au CNAM de Nouméa)

M. Jean CHEVEAU (directeur inter-régional des douanes de Rouen)

M. James VIGITELLO (inspecteur TSI à la direction inter-régionale des douanes de Rouen)

Du développement à la mise en production d'une application web : le projet MORSE (Module d'Organisation des Ressources Spécialisées et des Equipages)

RESUME :

La Direction Régionale Gardes-Côtes Manche Mer du Nord (DRGC MMN) a exprimé le besoin d'automatiser la gestion de ses personnels spécialisés marins et aériens et ses équipages.
La particularité de ce projet est la mise en place d'une organisation spécifique pour pallier la distance qui sépare la maîtrise d'ouvrage de la maîtrise d'œuvre.
Dans les entreprises privées, le télétravail est un domaine qui se développe mais qui reste encore rare.
Mais au sein d'une administration et particulièrement celle de la Direction Générale des Douanes et Droits Indirects (DGDDI) dont dépend la DRGC MMN, le télétravail pour mener à bien un projet informatique constitue sans doute une première.
Le mémoire présenté ici, retrace les différentes étapes du cycle de vie d'un projet et décrit plus en détails la phase de développement de l'application dénommée MORSE (Module d'Organisation des Ressources Spécialisées et des Equipages). Cette dernière a été développée avec les technologies web (AGL Eclipse, base de données Oracle 10g, JasperReports, ...) de nos jours incontournables pour une politique « tout internet ». Il dresse également un bilan du travail accompli en dehors des structures informatiques usuelles de la DGDDI.

Mot-clés : AGL, connexion RPC, DGDDI, DI de Rouen, Douane, DRGC MMN, personnels spécialisés, « tout internet »

ABSTRACT :

The Regional Coastguard Directorate of Manche Mer du Nord (DRGC MMN) wished to develop a software to manage its specialized staff (sailors and aircraft pilots) and its crews.
This project required the setting-up of a specific framework because the contracting owner and the project manager where located far from each other.
Private companies tend to use more and more telecommuting (or telework).
However, within a public service, and particularly the French General Customs and Excise Duties Directorate (DGDDI), and which includes the DRGC MMN, it was the first time telecommuting was used to carry out a computerised project.
This report provides a description of the software development process of a project with a particular emphasize on the development activity. The project is called MORSE (Module d'Organisation des Ressources Spécialisées et des Equipages). The software has been developped with web technologies such as IDE Eclipse, database Oracle 10g, iReports for JasperReports, ... which are current IT (Information Technologies) standards.
This report also gives a stocktake of all the job carried out outside the normal framework of the DGDDI's computers centers.

Keywords : IDE, RPC, DGDDI, DI de Rouen, Customs, DRGC MMN, specialised staff, « web technologies »

Penser, c'est morceler en idées nettes et incompatibles le réel complexe,

dont l'essence est l'indivisibilité concrète.

Liou Kia-Hway

REMERCIEMENTS

Je tiens à remercier tout particulièrement M. Jean CHEVEAU, directeur inter-régional de Rouen (et qui fût aussi mon directeur régional à Nouméa de 2003 à 2006), sans qui, ce projet n'aurait jamais vu le jour.
Je le remercie pour toute l'attention et la confiance qu'il a su me porter durant mes études à l'EICNAM.
Je le remercie également pour avoir cru en mes compétences pour mener à bien le projet MORSE.

Je remercie la DRGC MMN et tout particulièrement M. Eric DUPONT-DUTILLOY qui était directeur de la DRGC MMN au démarrage du projet ; M. Guillaume PAPE ; M. Vincent GUIVARCH ; M. Pascal TOTAL ; M. DIONET, directeur de la DRGC MMN ; Mme Sabine BENEDE ; M. Fabrice AUGNET ; M. Yannick PISANI, pour leur grande patience et leur soutien tout au long du projet et M. James VIGITELLO, inspecteur TSI à la DI de Rouen pour sa précieuse aide technique.

Je remercie M. Jean-Claude CAZALBOU, directeur adjoint du Centre Informatique Douanier à Osny et qui a été mon chef de projet pour SYDONIA, pour tous les précieux renseignements qu'il a pu me fournir.

Je remercie mon collègue et ami Rudolph FICADIERE, pour toute l'aide qu'il a pu m'apporter malgré son programme très chargé de PR.

Je remercie mes acolytes de l'EICNAM à Nouméa, en particulier, Noël CLERAMBOURG et Brigitte HUBERT, pour tout le soutien moral et l'entraide nécessaires à l'entreprise d'un tel projet que représente le diplôme d'ingénieur.

Je remercie également le CRA CNAM de Nouméa, M. Bernard SCHALL pour son éternel optimisme qui le fait croire en ses élèves, Nathalie POLI pour son soutien dans nos démarches administratives, ô combien fastidieuses auprès de l'EICNAM, sans oublier Joëlle LILO pour sa gentillesse lors des surveillances d'examens.

Je remercie Henri LAUGIE, mon tuteur CNAM pour ce mémoire d'ingénieur.

Je remercie les professeurs de l'EICNAM qui malgré la distance nous ont gentiment conseillés, le professeur Jacky AKOKA, M. Marc HIMBERT, M. Claude GENIER,

Je remercie Jacques THILLIER, qui a participé en tant que membre du jury à ma soutenance.

Je remercie très très chaleureusement mon collègue et ami Fabrice DANG pour toute sa patience et sa pédagogie pour m'apprendre à programmer comme lui, enfin presque ! L'élève ne dépassera pas le maître . . .

Je remercie mon ami Philippe DANG qui m'a dit "nos ancêtres ont suffisamment courbé l'échine, il nous appartient de relever la tête et de rester dignes pour eux" et mon ami Guy AGNIEL qui m'a beaucoup aidée.

Je n'oublie pas de remercier mon mari Jean-François qui a eu la patience de me supporter tout au long de ce cycle d'ingénieur et je remercie aussi mes enfants Duy et My-Lan qui ont eu suffisamment d'amour pour laisser leur maman travailler.

Je remercie aussi mes parents pour m'avoir aidée à m'occuper de mes enfants tout au long de mon projet et mes sœurs Marina et Corine qui ont cru en leur ainée.

Je remercie mes collègues et mes amis que j'ai négligés pendant quelques mois et qui ont toujours cru en moi, Jacques et Bernadette, Lisette, Cécile, Popo, Christine, Michèle, Astrid et Patrick, Jean-Marie, Michel, Andrée, Myriam, Chantal, Huguette, Agnès, Dominique, Jéromine, Isabelle dite "la petite", Tachou, Kamelou "mon p'tit frère", Nicolas, Magali, JSK, la consulaire, Manu, Mumble, Thierry, Patrick, Valérie, Julien, Coko.

Je remercie enfin, tous ceux qui de loin ou de près m'ont soutenue dans ce périlleux projet de diplôme d'ingénieur.

Table des matières

1. INTRODUCTION

La Direction Générale des Douanes et Droits Indirects (DGDDI)[1] est une direction du ministère du budget, des comptes publics et de la réforme de l'Etat.

Dès les années 70, elle a fortement investi dans l'informatique avec le projet SOFI[2], téléservice[3] d'établissement des déclarations en douane.

Comme toutes les administrations de l'Etat, après une première évolution liée à l'apparition de la microinformatique , l'émergence des NTIC[4] a été l'occasion d'une profonde remise en cause de ses infrastructures et méthodes de travail ainsi qu'une rapide extension des moyens et applications à l'ensemble de ses domaines d'activité.

Une des sous-directions de l'administration centrale et deux centres de calcul sont en charge de l'évolution et de la maintenance du système d'information et de télécommunication.

En 2009, la DGDDI comptait 18 000 agents dont plus de 300 informaticiens (hors assistance de proximité et centres de saisie) qui se répartissent entre les différentes qualifications informatiques répertoriées au sein du ministère.

Les limites des ressources disponibles et la priorisation de chantiers ne peuvent pas toujours permettre de satisfaire tous les besoins spécifiques à des populations réduites d'utilisateurs ou limités à des sujets d'enjeu relativement réduit. Ainsi, dans le cadre de ce mémoire d'ingénieur, il m'a été proposé de mener à bien un projet de gestion des équipages et des personnels spécialisés de la Direction Régionale des Gardes Côtes Manche Mer du Nord (DRGC-MMN), une des composantes de la Direction Inter régionale de Rouen (DI Rouen).

Ce projet est dénommé MORSE (Module d'Organisation des Ressources Spécialisées et des Equipages).

La démarche de conduite du projet MORSE, ne se distingue pas de la norme, elle a toutefois été plus complexe à mettre en œuvre parce que ce projet a été mené à distance, la maîtrise d'ouvrage (MOA) se trouvant éloignée de la maîtrise d'œuvre (MOE), il a donc fallu mettre en place des méthodes de travail atypiques tel que le télétravail.

Toute l'analyse des besoins et de la solution préconisée a été menée en amont de ce mémoire d'ingénieur. Le développement de l'application en programmation objet et son déploiement sont abordés en détails dans le cadre de ce mémoire.

---

1 Voir organigramme de la DGDDI en annexe 1

2 SOFI : Système d'Ordinateurs pour le Fret International

3 dénomination des applications web au sein de la DGDDI

4 NTIC : Nouvelles Technologies de l'Information et de la Communication

En tant qu'inspectrice analyste de la DGDDI, mes connaissances en conduite de projet m'ont permis d'aborder la phase d'analyse sans problèmes majeurs. Mes connaissances en développement objet étant plus restreintes, je me suis investie dans des recherches approfondies notamment sur les Ateliers de Génie Logiciel (AGL) tel que Eclipse[5] mais aussi sur la programmation objet.

Le temps imparti et l'unicité de la ressource m'ont amenée à ré-orienter mon travail sur ce projet en faisant appel à un programmeur expérimenté pour obtenir une formation plus rapide en programmation liée à l'accès base de données et finir l'application commandée.
Suite à des tests concluants au sein de la DRGC-MMN, le déploiement de cette application à d'autres directions nécessitera une adaptation afin d'intégrer les référentiels douaniers existants tel que RUSH[6].

[5] AGL développé par Eclipse Foundation dont la dernière version est GALILEO 3.5 (24/06/09)

[6] RUSH : Référentiel des Utilisateurs, des Services et des Habilitations

2. L'INFORMATIQUE AU SEIN DE LA DGDDI

2.1. Les services informatiques

Ils sont essentiellement regroupés au sein d'une des six sous-directions de l'administration centrale et dans deux centres de calcul. Des cellules techniques assurant un service de proximité auprès des utilisateurs sont en outre implantées auprès de chaque direction régionale des douanes.

2.1.1. La sous-direction C[7] : Systèmes d'information et de télécommunication

Elle est chargée de déterminer la politique technique de développement et d'exploitation informatique et de définir l'organisation des deux centres de calcul dont elle coordonne l'activité (Centre Informatique Douanier – CID[8] à Cergy-Pontoise et Direction Nationale des Statistiques et du Commerce Extérieur – DNSCE[9] à Toulouse). Elle assiste les maîtres d'ouvrage dans leurs études de besoin et veille à la satisfaction des utilisateurs du système d'information et de télécommunication.

2.1.1.1. Le bureau C/1 : Etudes et projets du système d'information

Il est chargé de définir le plus en amont possible les projets d'évolutions fonctionnelles du système d'information (SI) et d'assister les maîtres d'ouvrage dans l'analyse des besoins. Il garantit la cohérence fonctionnelle du SI en validant l'architecture fonctionnelle de tout projet à son lancement. De plus, il contribue à la normalisation des échanges avec les opérateurs de l'Union européenne.

2.1.1.2. Le bureau C/2 : Architecture technique et sécurité

Il définit la politique technique de développement et d'exploitation, notamment dans le cadre des orientations ministérielles et s'assure du respect de celles-ci au lancement de chaque projet. Il assure la maîtrise d'ouvrage déléguée des travaux de construction des équipements informatiques et réseaux communs et coordonne l'activité des centres de calcul pour les aspects relevant du domaine de l'informatique et des télécommunications.

---

7 Voir organigramme de la sous-direction C en annexe 2

8 Voir organigramme du CID en annexe 3

9 Voir organigramme de la DNSCE en annexe 4

2.1.1.3. Le bureau C/3 : Soutien et satisfaction des utilisateurs

Il assure le soutien aux utilisateurs en pilotant et animant le réseau informatique douanier dans chaque région à travers les cellules TSI (Techniciens des Systèmes d'Information). Il a la charge des tâches de supports au bénéfice de l'informatique et participe à ce titre à la préparation et au suivi des budgets informatiques et pilote l'adaptation des compétences des équipes informatiques et des autres services de la DGDDI en charge de ces domaines.

2.1.2. Le CID

Le CID, a un rôle de conception, de développement et de mise en production des projets informatiques tout en assurant la formation, l'assistance aux utilisateurs et la maintenance applicative.

Il est composé de 70% d'informaticiens (fonctionnaires de l'Etat ayant une qualification informatique, pupitreur, programmeur, analyste, chef de projet) et de 30% de personnel administratif.

Il est réparti en 4 divisions : développement 1 et 2, fonctionnement et services transverses.

Les deux divisions de développement ont en charge la conception, la réalisation, les tests et la maintenance applicative sur gros ou micro-systèmes.

La division du fonctionnement est en charge de l'exploitation des équipements centraux, de la surveillance et de l'administration des réseaux, de l'information et de la formation des utilisateurs, de la gestion des contrats relatifs aux équipements et aux liaisons de télécommunication et de la sécurité informatique.

La division des services transverses administre les bases de données, met les logiciels en production, s'assure du contrôle de la qualité des travaux réalisés et gère les ressources du centre informatique, du parc informatique et de la documentation.

Le CID assure la maintenance des téléprocédures et téléservices qu'il a développés en nouvelles technologies (ex : DELTA[10]) ou qui l'ont été ailleurs, en administration centrale ou par des Sociétés de Services en Ingénierie Informatique (SSII) dans le cadre des marchés. La maintenance des applications historiques mais également

[10] DELTA : Dédouanement En Ligne par Transaction Automatisée

vieillissantes et désormais hors palier, a été externalisée au travers de la mise en place de Tierces Maintenances Applicatives (TMA) pilotées par le Centre.

2.1.3. La DNSCE

La fonction initiale de la DNSCE, était la production statistique : « chiffre » du commerce extérieur à partir des données collectées au niveau des opérations de dédouanement, qu'elles soient fournies par le CID (déclarations établies dans le système SOFI puis DELTA), par les opérateurs eux-mêmes, notamment ceux établissant leurs déclarations par leurs propres moyens dans le cadre des procédures spécifiques et enfin par les quatre Centres Inter-régionaux de Saisie des Données (CISD) pour les échanges intra-communautaires.
Elle produit et exploite un nombre croissant d'applications dans d'autres domaines pour satisfaire les besoins des services.

La DNSCE assure la gestion de l'exploitation et l'administration des services mis à disposition des entités administratives internes et externes. Elle dispose d'une architecture de type mainframe pour les traitements en différé (MVS/VM[11]) et de type client-serveur pour les services internet.
La DNSCE est centre de compétence LINUX et administre les bases de données et les moteurs de recherches.

2.1.4. Les cellules de Techniciens des Systèmes d'Information (TSI)

Réparties sur l'ensemble du territoire et chargées du soutien des utilisateurs internes en matière de bureautique, les cellules TSI constituent après les services d'assistance aux utilisateurs des centres de calcul centrés sur l'emploi tant interne qu'externe des téléservices, le second axe d'assistance aux utilisateurs douaniers.
La mission de la cellule TSI porte essentiellement sur l'assistance aux utilisateurs et sur le dépannage de premier niveau des matériels même si d'autres tâches couvrant notamment les domaines de la gestion et de la formation peuvent faire partie de ses attributions.
Dans les faits, les tâches liées à la gestion de parc, comprises au sens le plus large sont importantes et divers constats sur l'évolution des fonctions autant que les

[11] MVS/VM : Multiple Virtual Storage/Virtual Machine

comparaisons faites avec d'autres structures tendent à envisager une évolution fondée notamment sur une plus grande proportion de résolution à distance des demandes d'assistance.

2.2. Les ressources informatiques

A ses débuts, du fait de la pénurie de compétences en la matière, la douane, comme les autres services de l'Etat a du former des informaticiens sur ses propres ressources. La loi du 23 décembre 1970[12] a organisé ces fonctions nouvelles dans l'administration. Elle est toujours le texte de référence, même si les conditions d'exercice, l'environnement de production et les techniques mises en œuvre ont considérablement évolués.

Le développement de la ressource, sa professionnalisation comme la diversification des compétences ou la politique de l'Administration ont modifié les conditions d'appel aux compétences.

Celles-ci ont deux sources :

- Interne :
  - les agents fonctionnaires de l'Etat, recrutés par concours ont pour la plupart acquis des qualifications informatiques dans l'enseignement supérieur. Ils constituent la majorité des personnels employés par l'administration ;
  - les agents contractuels, recrutés pour une durée déterminée, ils apportent les compétences qualitatives ou quantitatives, qui a un moment déterminé, font défaut.
- Externe : ce sont les Sociétés de Services d'Ingénierie Informatique (SSII) qui dans le cadre d'un projet dont la réalisation partielle ou totale (y compris de tierce maintenance applicative ou de tierce exploitation) interviennent dans le cadre d'un appel d'offre.

2.3. Le cycle de vie d'un projet : point de vue de l'administration

Sur une définition classique de projet : « un projet est un effort limité dans le temps pour créer un nouveau produit et/ou service qui crée de la valeur pour l'entreprise », on distingue une notion de temps (un projet a un début et une fin), une notion d'unicité (un projet s'oppose à une action répétitive, il est unique), une notion de ressources (les moyens mis en œuvre en terme de ressources humaines, compétences et matériels) et une notion de planification/régulation.

[12] Voir textes en annexe 5 et qualifications informatiques en annexe 6

Basé sur un cycle de vie de projet standard, la DGDDI a défini un cadre de projet qui reprend la conception, la réalisation et l'expérimentation.

Ainsi, elle pose des critères de durée de projet de « taille maîtrisable » qui vont de 1 à 12 mois et qui incluent la planification estimée à une durée maximum de 1 mois.

Tout projet a des objectifs, des acteurs, des processus et une fin.

Dans les processus, on distingue « le lancement du projet », « la conception », « la réalisation » et l'expérimentation[13].

2.3.1. Les objectifs

La phase projet doit permettre, après une période d'expérimentation, de valider sur un nombre limité d'utilisateurs l'atteinte des objectifs de la réalisation informatique prévue par le rapport d'étude.

2.3.2. Les acteurs

La MOA est représentée par le bureau réglementaire et le bureau C/2 pour les projets techniques.

La MOE est composée d'un chef de projet et d'une équipe de projet en charge de la conception et du développement. Elle est représentée par les centres informatiques, le bureau C/1 ou le bureau C/2.

Les bureaux C/1 ou C/2 assurent l'AMOA[14].

Le chef de projet régional est une personne désignée au sein d'une direction pour assurer le lien entre les utilisateurs et la MOA.

2.3.3. Les processus

Un comité de pilotage resserré, est constitué, présidé par la MOA, la MOE (centre de calcul ou SSII), du bureau C/1 ou C/2, de responsables utilisateurs choisis par la MOA. Il se réunit en général mensuellement avec pour mission de bien synchroniser les multiples actions du projet et prendre les décisions utiles pour rester centré sur les objectifs et la faisabilité décrits dans le rapport d'étude.

Il examine tout particulièrement chaque mois l'avancement du projet de réalisation informatique présenté par le chef de projet MOE et synthétisé dans la Fiche

[13] Voir Cycle de vie d'un projet informatique à la DGDDI en annexe 7

[14] AMOA : « assistance à maîtrise d'ouvrage », elle aide la MOA à définir et exprimer ses besoins

d'Avancement de Projet (FAP), veille au traitement des risques, aléas et modifications intervenant au cours du projet et acte la validation des productions et expérimentation ponctuant le bon avancement du projet.

Les compte-rendus de comité de pilotage tracent et synthétisent les décisions prises et incluent un tableau de suivi des actions de synchronisation importantes à suivre au niveau de ce comité.

2.3.3.1. Le lancement du projet

Lorsque la MOE est désignée, en principe avant la fin de l'étude, il aide la MOA à décliner le cahier des charges fonctionnel à partir du rapport d'études et prépare le Plan Projet Succinct (PPS). Quatre semaines au plus tard après la fin d'études, ou la désignation des chefs de projet MOE et MOA, une réunion de lancement de projet est organisée par le comité de pilotage.

Cette réunion vise à obtenir un accord global des différents partenaires sur la déclinaison du rapport d'études en cahier des charges et PPS.

Suite à la réunion de lancement, le projet va alors suivre les trois étapes suivantes que sont la conception (maquettage d'écrans éventuel), la réalisation (prototypage de l'application éventuel) et l'expérimentation (définie sur un périmètre limité de sites pilotes). Ces étapes peuvent se recouvrir si la réalisation est effectuée par lots.

Les livrables de cette phase sont :

- **Le cahier des charges** : rédigé conjointement entre le chef de projet MOE et MOA, il permet de compléter le rapport d'étude, engage la MOA et facilite l'appropriation de l'équipe MOE. Chaque modification ultérieure entraîne une analyse d'impact par le MOE, notamment en regard du PPS, et fait l'objet d'une décision en comité de pilotage pour sa mise en œuvre effective.
- **Le Plan Projet Succinct (PPS)** : rédigé par le chef de projet, il présente les enjeux, les grandes étapes du projet et le calendrier prévisionnel. Ce document est validé par la MOA avec l'assistance du bureau C/1 et/ou C/2.
- **Une fiche cartographique** (provisoire à ce stade) est livrée de manière concomitante par le chef de projet pour s'assurer de la bonne compréhension mututelle de l'objectif à atteindre.

2.3.3.2. La conception

Le chef de projet recense de façon exhaustive les aspects fonctionnels et techniques de l'application, afin de décrire précisément les besoins de la MOA. L'objectif est de limiter les incompréhensions fonctionnelles entre la MOA et le MOE lors de la livraison de l'application.

Les livrables de cette phase sont :

- **Les spécifications fonctionnelles, générales puis détaillées** (rédigées par l'équipe de MOE), qui décrivent les fonctionnalités du système dont seulement les règles de gestion sont validés par la MOA, avec l'assistance du bureau C/1 en cas de besoin.
- **Le dossier d'architecture technique** explicitant la solution technique, les éléments de dimensionnement du système et les principales exigences d'exploitation. Ce document, est obligatoire pour chaque projet et le bureau C/2 doit valider sa conformité au palier technique « Tout Internet » de la Douane.

2.3.3.3. La réalisation

L'équipe projet réalise les spécifications techniques détaillées, documents internes à la MOE. Les opérations de codage proprement dit peuvent alors débuter.

En parallèle, la MOA doit élaborer les jeux d'essais fonctionnels (tests à effectuer), avec la collaboration de la MOE lui permettant de vérifier la conformité du développement.

Elle doit également assurer la sensibilisation des utilisateurs aux changements éventuellement introduits par le projet et concevoir la phase de formation et les supports nécessaires (dont manuel utilisateurs). La MOA a la responsabilité d'organiser cette phase et les documents nécessaires, avec l'aide de la MOE, des bureaux C ou des services de formation de la douane.

Les livrables de cette phase sont :

- **Les jeux d'essais** (tests à effectuer) réalisés par la MOA en collaboration avec la MOE.
- **Le guide utilisateur** (ou une aide en ligne) est élaboré sous la responsabilité du chef de projet MOA (en collaboration avec des utilisateurs et la MOE).
- **Les sources et exécutables** de l'application et la documentation associée.

2.3.3.4. L'expérimentation

Selon le projet, l'expérimentation a une durée approximative de quelques semaines à trois mois.

Cette phase a pour objectif de mettre l'application, qui à ce stade est considérée comme achevée, du point de vue des utilisateurs, à la disposition de quelques sites pilotes incluant les utilisateurs investis dans la co-construction et quelques nouveaux services utilisateurs en nombre limité. Ils devront valider : le caractère opérationnel du produit logiciel, du dispositif de formation, de la documentation utilisateurs et des conditions du déploiement.

Pour les projets concernant un grand nombre d'utilisateurs, le Service d'Assistance Utilisateur (SAU) du centre informatique réalisateur doit être associé à cette phase pour préparer le soutien des utilisateurs à l'emploi de ce nouvel applicatif. Le bureau C/3 doit préparer la prise en charge dans le système de soutien Traitement et Suivi de l'Assistance en Réseau (TSAR) de la nouvelle application généralisée en y associant, pour chacune d'entre-elles, un pôle d'assistance de niveau 1 (assistance immédiate par l'agent TSI) et 2 (problème transmis à un niveau supérieur, centre de calcul par exemple), selon les modalités convenues avec le bureau MOA pour le soutien de niveau 2.

Compte tenu de l'objectif de validation sur un nombre restreint d'utilisateurs, les opérations techniques de mise en production, automatisation de la production et tenue en charge ne sont pas obligatoirement terminées pour lancer cette phase et peuvent s'achever en parallèle.

L'expérimentation est coordonnée par le bureau C/3 et s'organise en trois étapes successives :

- le comité de pilotage effectue le lancement de cette phase après désignation des sites expérimentateurs par la MOA

➢la réunion d'étape permet l'examen des fiches d'appréciation des sites pilotes de l'application livrée avec les participants cités précédemment

➢le comité de pilotage doit constater avec les chefs de projet régionaux des sites pilotes et le bureau C/3 la bonne maîtrise de la phase d'expérimentation et en prononcer le bilan.

Les livrables de cette phase sont :

●**Les compte-rendus des jalons d'expérimentation** issus du comité de pilotage du projet intégrant la hiérarchie des sites pilotes.

●**Les fiches d'appréciation synthétiques** (bilan d'expérimentation) établies par les sites expérimentateurs avec avis obligatoire des directeurs pilotes.

●**La fiche cartographique** actualisée par le chef de projet MOE avant la fin du projet et transmise au bureau C/3 pour publication sur intranet.

2.3.4. La fin de projet

Lorsque le bilan est déclaré positif par le comité de pilotage, une lettre de fin de projet établie par la sous-direction C, adressée aux MOA, MOE, bureaux C/1 et C/2 et directeurs pilotes, marque la fin du projet.

Une note de généralisation signée par la MOA est ensuite adressée aux services utilisateurs, rappelant les objectifs du projet, l'accompagnement prévu et le calendrier de déploiement. Lorsque le nombre d'utilisateurs est conséquent, le déploiement peut supposer une phase d'ingénierie de déploiement à part entière, organisée sous l'autorité de la MOA.

2.4. Le système d'information de la DGDDI

Le dédouanement, la fiscalité indirecte, la comptabilité, la lutte contre la fraude, le pilotage et les statistiques constituent les six grands domaines métiers du système d'information de la douane[15]. Différents référentiels (cf § 2.4.1 ci-dessous) fournissent aux téléservices, les informations communes nécessaires. Le système d'information est un espace sécurisé et protégé, dont l'accès est strictement limité aux personnes et aux systèmes habilités. Les douaniers y accèdent via Aladin[16], les opérateurs économiques via ProDou@ne et les systèmes d'informations partenaires via des connecteurs.

[15] Voir Système d'Information de la DGDDI en annexe 8
[16] Intranet de la DGDDI

2.4.1. Les référentiels

●RUSH : Le Référentiel des Utilisateurs, des Services et des Habilitations continuera à s'adapter à l'évolution organisationnelle de la douane et au futur Système d'Information des Ressources Humaines (SIRH) pour maîtriser et encore mieux définir les processus de gestion des identités.

●ROSA : Le Référentiel des Opérateurs et de Suivi des Agréments continuera à s'enrichir de nouvelles relations.

●RITA : Le Référentiel Intégré Tarifaire Automatisé recueille l'ensemble des exigences communautaires et nationales applicables en matière douanière aux marchandises faisant l'objet d'échanges internationaux.

●DROP : Le référentiel des Données de Référence Opérationnelles Partagées a pour vocation de définir et partager les tables de références communes à l'ensemble des téléservices (codes pays, codes postaux, codes taxe, monnaies, etc.).

●RAC : Sur le modèle de DROP, le Référentiel pour les Applications Communautaires a été mis à la disposition des téléservices en 2009.

2.4.2. Les connecteurs

Le système d'information (SI) de la DGDDI est conçu pour pouvoir gérer les échanges de données - parfois en temps réel - avec des applications extérieures hétérogènes. L'interconnexion des SI se fait à l'aide de connecteurs assurant une interface entre le système d'information et l'extérieur.

●Interface comptable pour le cycle de la recette (InterCOM)

InterCOM a vocation à prendre en compte l'ensemble des transactions comptables de la douane. Plus tard, InterCOM permettra de transférer les informations au système de comptabilité ministériel CHORUS recettes.

●Interface comptable pour le cycle de la dépense (InterDEP)

InterDEP permettra au système d'information douanier d'échanger des informations avec le système de comptabilité CHORUS dépenses. Elle sera appelée par tous les nouveaux téléservices qui nécessiteront une entrée en comptabilité dépense.

●Interface statistique

Un certain nombre de données issues des déclarations (DAU[17], DEB[18]...) sont exportées vers la chaîne statistique, pour constituer le « chiffre du commerce extérieur ». Afin de fiabiliser le processus, une nouvelle interface va être créée pour permettre à chaque téléservice d'exporter ses données de manière normalisée.

●Interface d'échanges des déclarations communautaires

Avec la montée en charge des programmes communautaires, les échanges de messages déclaratifs entre administrations douanières des Etats membres vont s'intensifier. Dans cette perspective, un nouveau module générique, nommé InterDOUANE, sera construit pour permettre à tous les systèmes nationaux d'adopter un format d'échange standard.

●Dispositif d'accès sécurisé à une application externe

Le dispositif **HAPEX** permettra aux douaniers d'accéder à certains téléservices d'administrations partenaires et inversement. Ce système sera basé sur la confiance réciproque des systèmes d'information partenaires.

2.4.3. Le résultat de l'urbanisation

Tout système d'information doit faire l'objet d'une démarche d'urbanisation et la DGDDI avec le grand nombre d'applications et de bases de données parfois redondantes accumulées au fil des ans, n'a pas échappé à cette démarche.

Afin d'urbaniser, il faut se poser certaines questions fondamentales : quels sont les principaux quartiers, comment les desservir, quelles données doivent circuler, à quel moment, quel chemin doivent-elles emprunter ?

[17] DAU : Document Administratif Unique

[18] DEB : Déclaration d'Echange de Biens

Ainsi on définit les grandes tendances de développement du SI avec une interconnexion simple entre les applications et un accès facilité aux données grâce aux référentiels.

L'urbanisation repose sur une connaissance actualisée des processus métiers, des besoins fonctionnels, et de l'environnement douanier.

Le concept de transversalité est indissociable de l'urbanisation (chaque application nouvelle est envisagée comme la partie d'un tout et non comme un bloc isolé et indépendant) rendant le SI plus modulaire et plus réactif tout en facilitant l'adaptation du système d'information aux nouveaux enjeux

L'objectif principal de l'urbanisation est la capacité permanente à aligner le système d'information à la stratégie de la Douane évoluant dans un cadre communautaire, interministériel, ministériel et directionnel.

La DGDDI a donc entrepris cette démarche d'urbanisation en 2001 (voir le schéma du SI en annexe 8). Si le socle d'urbanisation est désormais posé (les référentiels, les connecteurs, les quartiers, etc ...), le chantier est encore ouvert et soumis aux évolutions « tout internet » des applications métiers. La DGDDI s'attèle donc à ré-écrire les téléservices en déclinant une Trajectoire d'Evolution du SI (TESI) pour parvenir à une urbanisation efficace et complète de son SI. Les grandes lignes d'action et les priorités sont déclinées ci-dessous.

	2009			2010	2011
Priorité	Opération (1)	Complexité	MOA	Opération	Opération
1	eCUSTOMS - Amendt sécurité - ICS/ECS/NSTI/DELT@	👆👆👆👆	**E3**-D2-D3	*ECUSTOMS AES/AIS*	*ECUSTOMS AES/AIS*
	eCUSTOMS - OEA (DELT@/ROSA accréditation)	👆👆	E3	*CHORUS recettes*	*ECUSTOMS Single Windows*
	eCUSTOMS - EORI (ROSA)	👆	E3	*ARIANE (SIRHius)*	*CHORUS recettes*
	eCUSTOMS - TARIC 3 (RITA)	👆👆	E4	*TPL*	*ARIANE (SIRHius)*
	eCUSTOMS - refonte Procédure Recherche (NSTI)	👆👆	E3	*Décisionnel SIDD (BANACO vision opérateur 360°)*	*TPL*
	EMCS – communautaire et EDI (GAMMA V2)	👆👆	F3	*Décisionnel SIDD (Analyse risque dédouanement)*	*Décisionnel SIDD (LCF)*
	ARIANE (SIRHius)	👆👆👆👆	CS	*Décisionnel SIDD (ARIANE)*	*Décisionnel SIDD (CI)*
	CHORUS dépenses	👆	B1	*ALADIN NG*	
	Compta auxiliaire operateurs (INTERCOM)	👆👆	B1		
	TGAP nvlles composantes et télé-procédure	👆	F2		
	TPL Taxe Poids Lourds	👆👆	F1		
	Base Nationale Contrôle (BANACO) fiche contrôle	👆👆	D2		
	DEMATERIALISATION (7 sujets bottom up)	👆👆	C		
	ALADIN NG dont refonte Courrier et Dana	👆👆	Cab		
	DRM (GILDA)	👆	B1		
	Décisionnel SIDD (référentiels et tableau de bord DG)	👆👆👆👆	Cab		
2	NCVI - Nouveau Casier Viticole	👆👆👆	F3	*Stat CE : évolutions Extrastat*	*ALADIN NG : Refonte ProDouane*
	DCI - Dispositif Continuité Informatique	👆👆👆👆	C	*TIPP DOM (ISOPE)*	
	Détaxe (PABLO déploiement)	👆	F1	*DCI - Dispositif Continuité Informatique*	
	Stat CE : DEB DTI+	👆	DSEE	*Stat CE : assouplir coeur de chaîne*	
	Stat CE : CORINTHE DAU2007	👆👆	DSEE	*Détaxe commerçants occasionnels (PABLO)*	
	Stat CE : INES production souples	👆	DSEE	*RCO Renseigt contraignant origine*	
	LAPI : Lecteur plaques automatisés	👆	D3	*Avis valeur douane*	
	TIPP Reg avitaillement (ISOPE)	👆	F2	*ALADIN NG : Refonte portail mobile PODIUM*	
	DES Déclaration Echange de Services ??	👆👆	DSEE	*DELT@T : intégration NSTI*	
	Contrôle interne référentiels RUSH / ROSA	👆	C	*DELT@ occasionnels*	
	Référentiel véhicules	👆👆	C	*Référentiel véhicules*	
	Téléprocédure Didou	👆	E4		
	Téléprocédure Dani	👆	**F1**-B1		
3	PCI	👆👆	D1	*Téléprocédure Rider*	
	Visabio	👆	A3	*Téléprocédure Rdt*	
	TSVR (évolutions)	👆	**F1**-B1		
	TIPP Remboursement SIDECAR (évolutions)	👆	F2		
	Véhicules diplomatiques	👆	F1		
	Stat CE : Astrinet V3 / Info TVA	👆	DSEE		
	Infos perturbations sociales	👆	A1		
	Gestion BOP	👆	CCG		

Métier communautaire	👆 = simple
Métier interministériel	👆👆 = moyen
Métier directionnel	👆👆👆 = important
Support (inter)ministériel	👆👆👆👆 = large
Support directionnel	

3. L'ÉTUDE REALISÉE POUR CE MÉMOIRE

3.1. Préambule : le projet MORSE

3.1.1. Le contexte

La nécessité de disposer d'un projet qui s'inscrive dans le cadre du diplôme d'ingénieur m'a conduite à interroger différentes directions de la DGDDI sur un besoin éventuel.

La Direction Inter-régionale de Rouen (DI de Rouen) a répondu favorablement à ma demande en me proposant un projet qui puisse satisfaire un besoin spécifique exprimé par la Direction Régionale des Gardes Côtes Manche Mer du Nord (DRGC-MMN).

En poste en Nouvelle-Calédonie, le premier écueil de ce projet a été la distance qui sépare ma direction de celle qui me proposait ce travail.

La Direction Régionale des Douanes de Nouvelle-Calédonie (DRDNC), où je suis affectée en tant qu'inspectrice analyste, est une des directions régionales de la DGDDI. Elle a pour particularité de mener ses missions pour le compte du Gouvernement de la Nouvelle-Calédonie (GNC).

Ainsi, ses agents administrativement gérés par la DGDDI sont mis à disposition du GNC.

Il était de ce fait évident qu'une méthode non éprouvée allait devoir se mettre en place afin de mener à bien le projet qui m'était confié.

Le tuteur « entreprise » choisi pour m'assister sur ce mémoire de fin de cycle d'ingénieur est monsieur Jean CHEVEAU, directeur inter-régional de la DI de Rouen.

Le tuteur CNAM[19] désigné est monsieur Henri LAUGIE, enseignant du CNAM Nouméa.

3.1.2. Qualifications et expériences professionnelles

Entrée dans la fonction publique d'Etat en 2000 par concours de catégorie C, je disposais d'un diplôme universitaire australien équivalent à un DUT[20] informatique. Je ne possédais alors aucune qualification informatique administrative décrite ci-dessus.

[19] CNAM : Conservatoire National des Arts et Métiers

[20] DUT : Diplôme Universitaire de Technologie

Je présentais le premier examen de PAU en 2001 tout en présentant un concours de catégorie supérieure avec qualification « programmeur ». J'ai ensuite réussi un concours de catégorie A avec la qualification « analyste » qui m'a permis d'occuper un poste informatique au sein d'une cellule TSI. J'ai par ailleurs entrepris des études au CNAM de Nouméa pour obtenir en 2005 un DEST[21] informatique « ingénierie et intégration informatique : systèmes d'information ».
Les formations diplômantes ne sont pas « reconnues » par l'administration pour la prise en compte des fonctions exercées. Ainsi, il nous appartient de présenter les examens ouvrants aux qualifications informatiques de la fonction publique.
Je présenterai en fin d'année, l'examen à l'une des deux dernières qualifications qu'il m'est encore possible d'obtenir par rapport à ma catégorie : PSE-CRA.

Si j'ai pris en charge l'ensemble de ce projet, l'étude menée pour ce mémoire est essentiellement basée sur le développement et la mise en production.
À travers mon cursus étudiant et professionnel, j'ai programmé dans divers langages[22] en commençant par ceux de seconde génération tel que l'Assembleur, en passant par des langages « intermédiaires » (COBOL 3ème génération) jusqu'aux langages objet actuellement utilisés (ASP, JAVA, PHP 3ème génération).
Cependant pour le projet MORSE, mes connaissances de programmation n'étant plus que purement théoriques depuis ces cinq dernières années, il m'a été nécessaire de faire des recherches approfondies et de dérouler des tutoriels afin de pouvoir commencer à programmer.
Au cours de mes études pour l'obtention du DEST informatique en 2004, un module a été complètement consacré au langage JAVA, ainsi mes connaissances dans ce langage se limitent à ces études, je n'ai jamais eu l'occasion de programmer pour une application mise en production au sein de mon administration.
J'ai cependant programmé en « objet », en langage ASP, pour une autre application mise en production : « OEILLADES[23] ».
Pour finir l'application MORSE, mes connaissances et mes recherches sur les connexions en base de données ont dues être abondées par une formation auprès d'un

[21] DEST : Diplôme d'Enseignement Supérieur Technique

[22] Les générations de langages informatiques, voir annexe 9

[23] OEILLADES : Outils d'Envois Informatisés Lié au Logiciel d'Acquisition de Données d'Enquêtes Sphinx

programmeur expérimenté. Le temps passé à l'auto-apprentissage devenait trop important pour pouvoir respecter les délais qui avaient été au préalable déterminés.

Auparavant, il m'a fallu également acquérir des techniques d'installation de l'AGL choisi (Eclipse) et de la base de données utilisée (Oracle 10g). Elles seront décrites dans le chapitre 4 suivant.

3.1.3. La Direction Inter-régionale de Rouen (DI de Rouen)

La DI de Rouen, du nom de son siège, est une des douze Directions Inter-régionales territoriales de la DGDDI.

Elle groupe quatre directions régionales, Rouen, Le Havre, Basse Normandie et Garde côtes Manche Mer du Nord.

Comme toutes les DI, elle exerce le pilotage stratégique et assure les tâches de support aux directions opérationnelles, comptables, de personnel et matérielles au sein de la circonscription. Elle assure la coordination et la cohérence entre performance et allocation de ressources aux directions régionales opérationnelles.

Elle est organisée autour de pôles :

- budget opérationnel de programme et ressources humaines
- logistique et informatique, malgré sa dénomination, celui-ci ne gère que les moyens et la satisfaction des besoins des utilisateurs locaux en s'appuyant sur le TSI inter-régional.

3.1.4. Direction Régionale des Garde Côtes Manche Mer du Nord (DRGC-MMN)

La douane est la seule administration civile de l'Etat, présente en mer, à mettre en œuvre un dispositif complet de surveillance, combinant une composante navale et une composante aérienne. Ce dispositif maritime a aussi la particularité d'être étroitement lié au dispositif douanier de surveillance terrestre.

Dans un cadre de coordination renforcée, des moyens de l'Etat[24] , sous l'autorité des préfets maritimes, la douane participe activement aux missions de service public relevant de l'action de l'Etat En Mer (AEM), dans les domaines qui constituent son coeur de métier (lutte contre les trafics illicites) ou pour ceux dont elle dispose de

[24] Affaires Maritimes, la Marine Nationale, la Gendarmerie Départementale, la Gendarmerie Maritime, la Société Nationale de Sauvetage en Mer (SNSM)

moyens et de compétences particulières (rejets illicites, pollution en mer) mais aussi dans les autres secteurs de l'AEM (surveillance et contrôle des pêches, lutte contre l'immigration illégale par voie maritime, sureté maritime et portuaire).

La DGDDI dispose de quatre DRGC, sur chacune des trois façades maritimes métropolitaines : « Atlantique », « Manche Mer du Nord », « Méditerranée » et « Antilles-Guyane ».

Les moyens maritimes (en métropole et outre-mer) sont composés de 2 patrouilleurs garde-côtes, 19 vedettes garde-côtes, 16 vedettes de surveillance rapprochée, 2 annexes rapides, 3 bateaux écoles et 20 embarcations légères.
La flotte aérienne pour l'ensemble de la DGDDI est composé de 12 avions biturbine dont 2 avions Polmar spécialement équipés d'un système de télédétection des pollutions marines et de 5 hélicoptères.

Ces moyens sont armés par des personnels spécialisés en possession de qualifications soumises à revalidation ou recyclage périodique.
Ces ressources composent les équipages des moyens garde-côtes, lesquels répondent à des normes de composition précises.

Le siège de la DRGC-MMN est à Canteleu (Rouen). Elle possède 5 navires :

- ●1 patrouilleur garde-côtes de 43m basé à Boulogne sur Mer « Jacques Oudart Fourmentin »
- ●2 vedettes garde-côtes de 28m, une basée à Cherbourg « Vent d'Amont » et une à Dunkerque « Vent d'Aval »
- ●1 vedette de surveillance rapprochée basée à Granville « Vire »
- ●3 hélicoptères EC135 basés au Havre

Elle dispose de 149 agents spécialisés affectés au fonctionnement et au pilotage opérationnel de ces moyens.
La façade maritime Manche-Mer du Nord s'étend du Mont Saint Michel à la frontière Belge sur 870 km de côtes[25]. Sa zone d'action en mer est celle de la Préfecture maritime à Cherbourg.

[25] Zone d'action de la DRGC-MMN voir annexe 10

Les conditions météorologiques difficiles d'octobre à avril, la densité du trafic (600 à 800 bateaux par jour qui représente 20% du trafic mondial et un échange de passagers entre la Grande-Bretagne et la France de 70 000 passagers par jour) constituent les principales caractéristiques de cette zone.

La zone couverte par la DRGC-MMN comprend le Havre, premier port français pour conteneurs ; Dunkerque, premier port français pour le minerais et le charbon ; Boulogne sur Mer, premier port de pêche européen et Rouen, premier port fluvial et fluvial-maritime céréalier d'Europe.

3.1.5. L'opportunité

La DI de Rouen et la DRGC-MMN effectuent manuellement le suivi des qualifications de chacune des ressources afin de vérifier que celles-ci correspondent à l'état de la réglementation en vigueur et que les différentes validations périodiques sont bien effectuées dans les délais (stages de mise à niveau, visites médicales périodiques et obligatoires, etc ...).

Il convient en outre de vérifier que le nombre et la combinaison des différentes qualifications permettent de composer des équipages dans le respect des normes.

Enfin, les deux directions doivent être capables d'appréhender les mesures à prendre à titre prévisionnel pour conserver le caractère opérationnel des équipages (propositions de recrutement, d'affectations, etc ...) dès lors que celui-ci est susceptible d'être affecté par des départs (retraite, mutation, etc ...).

Le besoin exprimé par la DRGC-MMN étant très spécifique et n'ayant jusqu'alors pas fait l'objet d'une demande formalisée d'évolution du SI, mais correspondant à un besoin vivement ressenti localement, ce projet m'a été proposé comme susceptible de fournir un sujet constituant la base de mon mémoire d'ingénieur.

Le projet MORSE a ainsi débuté fin 2008 avec une date prévisionnelle d'achèvement en fin 2009.

3.2. L'étude et l'analyse

3.2.1. La conduite de projet

3.2.1.1. Présentation

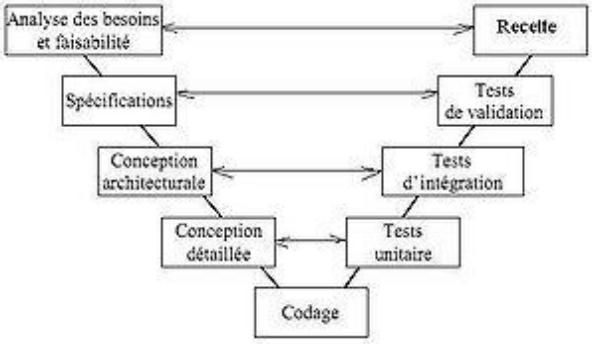
Un projet est un ensemble d'activités sur un temps défini pour répondre à un besoin.
Un projet implique toujours une nouveauté par rapport à l'existant.
Il se découpe en phases : études des besoins, analyse, réalisation, tests, recette et mise en production.
Un projet suit un cycle de vie choisi selon sa nature.

On distingue différents types de modèles de cycle de vie[26] :

- le modèle en cascade
- le modèle en V
- le modèle incrémental ou loti
- le modèle itératif
- le modèle en spirale
- la méthode de développement rapide d'applications (modèle RAD – Rapid Application Development)
- la méthode du prototypage
- la méthode DSDM (Dynamic Systems Development Method)
- la méthode RUP (Rational Unified Process)

Le **modèle en V** est une amélioration du modèle en cascade, il limite le retour à l'action précédente.
À chaque étape, un retour d'information est prévu pour l'action située en vis à vis sur le schéma du cycle en V.
Le cycle en V est le standard du développement logiciel et de la gestion de projet.

[26] Voir les différents modèles de cycle de vie en annexe 11



La gestion de projet permet de s'assurer que les actions menées respectent le délai imparti et le budget alloué tout en répondant aux besoins pour lesquels ce projet a été engendré.

Les ressources affectées au projet (humaines, matérielles, informationnelles, financières, ...) ont fait l'objet d'une budgétisation et seront gérées par le chef de projet désigné.

La gestion de projet suit des standards repris dans le PMBOK[27].

3.2.1.2. Le cahier des charges fonctionnel

Le cahier des charges fonctionnel (CDCF) est l'expression du besoin émis par la MOA. Il est élaboré selon certaines normes.

L'AFNOR[28] définit le CDCF comme « *un document par lequel le demandeur exprime son besoin (ou celui qu'il est chargé de traduire) en terme de fonctions de services et de contraintes. Pour chacune d'elles sont définis des critères d'appréciation et leurs niveaux. Chacun de ces niveaux doit être assorti d'une flexibilité.* »

La norme AFNOR concernant le CDCF est référencée NF X50-151[29], elle décrit un certain nombres de parties nécessaires à son élaboration :

- Une présentation générale du problème énonçant toutes les informations générales utiles sur le produit ou service à fournir.
- Une expression fonctionnelle des besoins qui décrit et définit les fonctions de service du produit, les contraintes, les critères d'appréciation…

[27] Project Management Body Of Knowledges, voir domaines du PMBOK en annexe 12

[28] AFNOR : Association Française de NORmalisation

[29] Plan type du CDFC selon la norme NF X50-151 de l'AFNOR en annexe 13

●Un appel à des variantes (optionnel) qui fixe des limites à l'étude ou à d'autre solutions possibles.

●Un cadre de réponse qui simplifie et codifie la façon de répondre.

Basé sur ce modèle, le CDCF du projet MORSE[30] a été élaboré par mes soins mais validé par la MOA qui ne possédait pas les compétences nécessaires pour la rédaction d'un cahier des charges fonctionnel. J'ai ainsi joué le rôle d'assistance à MOA.

Pour cela, trois réunions ont eu lieu en septembre, octobre et décembre 2008 à Rouen, afin de définir précisément les besoins de la MOA. De par la distance existante entre la MOA et la MOE (moi-même), ces réunions ont été les seules mises en place, le reste des échanges et validations ont été effectués au moyen de la messagerie électronique. Ainsi nous obtenons le cahier des charges fonctionnel version 3 (V 0.3), validé par la MOA le 8 décembre 2008.

L'objectif de cette assistance à MOA a été de lui permettre d'avoir *" une formulation exhaustive et aussi claire que possible de ce besoin. L'énoncé du besoin sous une forme fonctionnelle, c'est à dire en termes de finalités, sans référence aux solutions techniques susceptibles d'y répondre, préserve toutes les chances d'émergence de l'innovation au moment de sa conception. Cette formulation constitue l'expression fonctionnelle du besoin"* (NFX 50 151).

3.2.1.3. Le cahier des charges technique

Elaboré par la MOE, il décrit la solution envisagée sur un plan technique. La solution technique s'appuie sur la technologie physique (serveur, postes de travail, bases de données, etc ....) et la solution « fonctionnelle » préconisée (application, données, écrans, etc ....).

Le cahier des charges technique (CDCT) doit être validé par les deux parties (MOA et MOE) pour permettre de continuer le projet et mettre en œuvre la solution définie.

Il est la traduction des besoins fonctionnels décrits dans le CDCF en une solution concrète de produit ou service.

[30] CDCF du projet Morse voir en annexe 14

Le CDCT du projet MORSE[31] est composé de trois parties et a été validé par la MOA le 15 octobre 2009 dans sa version 3 (V 0.3).

Il décrit l'environnement de réalisation, l'organisation des données et l'implémentation des données.

Ce document a été rédigé selon un modèle de CDCT fourni par le CID et suivant les recommandations de ce dernier tout en respectant le palier technique[32] de la DGDDI.

Il constitue le premier livrable du projet car on considère que le cahier des charges fonctionnel est produit par la MOA.

Il a permis à la MOE de formaliser et proposer sa solution afin de continuer le projet MORSE dans sa phase de développement.

3.2.1.4. La planification du projet MORSE

Le détail de la planification des tâches sur le projet MORSE est repris dans le CDCT.

La charge de travail pour chacune des phases de développement a été définie selon le tableau ci-dessous :

Lot	Charge prévue	Ressource prévue	Particularité
1 - Agent	5 j/h	1 programmeur	
2 – Equipage	2 j/h	1 programmeur	
3 – Navire	2 j/h	1 programmeur	
4 – DRGC	1 j/h	1 programmeur	
5 – Règlementation	2 j/h	1 programmeur	
6 – Qualification	2 j/h	1 programmeur	
7 –Editions	5 j/h	1 programmeur	
8 – Paramétrage	1 j/h	1 programmeur	

Le développement a ainsi été découpé en modules afin de les rendre indépendants les uns des autres et permettre leur réalisation en parallèle ou en continu.

Chaque module correspond à une des fonctionnalités du projet MORSE.

[31] CDCT du projet Morse voire en annexe 15

[32] Palier technique 2007B de la DGDDI au 11/12/07 voir en annexe 16

La planification des phases du projet se décompose comme suit :

Phases	Du	Au	Observations
Développements	05/10/09	02/11/09	Par la MOE
Tests	09/11/09	10/11/09	Par la MOE
Formation	10/11/09	13/11/09	Par la MOE à distance
Recettage	16/11/09	22/11/09	Par la MOA à distance
Mise en Production	23/11/09	23/11/09	Uniquement pour la DRGC MMN
Evaluation/fin de projet	30/11/09	02/12/09	

Les différentes phases du projet présentées sont les phases classiques que l'on retrouve dans tout projet informatique. La phase de développement est la plus importante en terme de durée. On note une mise en production uniquement pour la DRGC MMN, la généralisation de l'application à d'autres DRGC n'entre pas dans le cadre de ce projet.

3.2.1.5. La conduite de projet MORSE

La conduite de projet sur MORSE, s'est effectuée sans problèmes majeurs sur les phases d'expression des besoins et la phase d'analyse.

Le modèle de conduite de projet choisie pour MORSE a été le modèle en cascades décrit ci-dessus, les validations ont été faîtes au fur et à mesure de l'avancée du projet avec validation de la MOA pour chaque livrable.

Aucun budget n'a été alloué à ce projet, les logiciels utilisés en phase de développement étant entièrement gratuits car issus de l'Open Source, ils sont cependant repris au palier technique de la DGDDI.

L'unicité de la ressource a entraîné des délais de livraison non négligeables.

L'application aurait due être livrée pour tests le 09 novembre 2009, elle ne sera finalement livrée pour tests qu'en septembre 2010.

Le planning décrit dans le paragraphe 3.2.1.4 ci-dessus n'a pu être respecté.

Dans le cadre d'un projet budgétisé, ces retards se traduiraient par une affectation supplémentaire de ressources pour pallier les délais induits. Mais cette action ne pourraient être envisagée sans considération d'un coût supplémentaire.

Pour le projet MORSE, la maîtrise des coûts ne peut être évoquée (absence de budget), la maîtrise des délais est intimement liée à la première.

Dans ce projet, la communication s'est essentiellement effectuée par mél, mettant ainsi l'implication des utilisateurs en péril.

La livraison pour tests sera un moyen de reprendre la communication entre la MOA et la MOE et d'amener à une conduite de changement qu'implique tout projet novateur.

En terme de processus métier, le retard pris se traduit pour les futurs utilisateurs par la poursuite de la gestion manuelle des personnels spécialisés toujours aussi fastidieuse et peu efficace.

Il est ainsi nécessaire de s'assurer que l'application soit livrée conformément aux besoins exprimés dans le CDCF.

La conduite de projet sur MORSE se poursuivra donc au delà de ce mémoire de fin de cycle d'ingénieur.

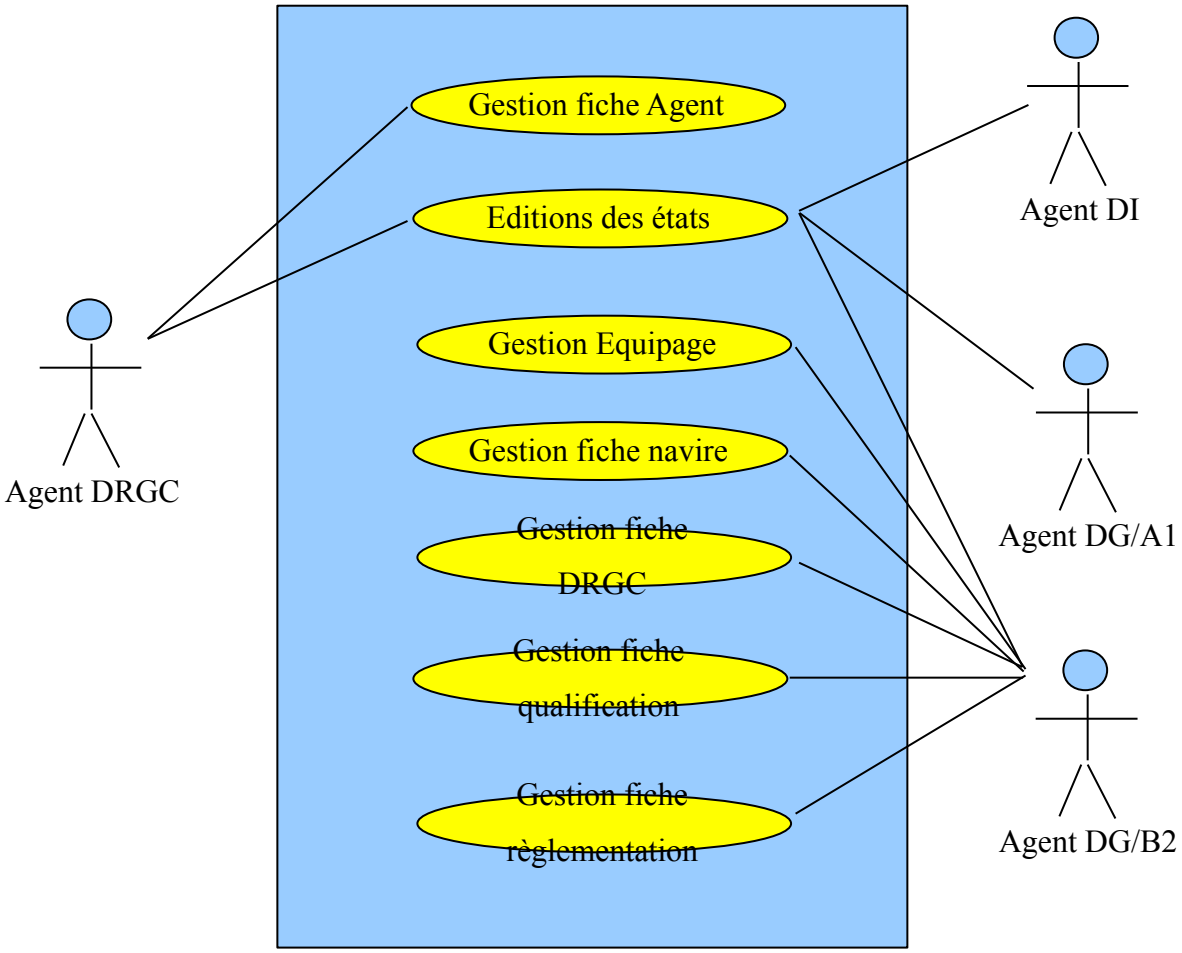
La recette de l'application marquera le terme du projet MORSE.

3.2.2. La solution envisagée

Le téléservice MORSE est dans un premier temps destiné à la DRGC-MMN, l'expérimentation par cette dernière sur une durée à déterminer permettra d'envisager ou non, son déploiement aux autres DRGC ou à d'autres services de la DGDDI qui le souhaiteraient.

Le diagramme de cas d'utilisation ci-dessous permet de voir les intéractions entre les différents acteurs du projet. Ce diagramme présente la solution finale (déploiement à toutes les DRGC et les services de la direction générale).

Dans le cadre du projet MORSE « restreint » à la DRGC MMN, « l'agent DRGC » est le seul acteur et accède à toutes les fonctionnalités.



`

La solution proposée pour ce projet repose sur :

- une architecture « poste client » avec les caractéristiques suivantes :
  - XHTML[33], version 1.0 et HTML[34] 4.01
  - Javascript selon la norme ECMA-Script, version 1.5
  - Navigateurs compatibles : Internet Explorer version 6 SP2 et Mozilla firefox version 3.
- une architecture serveur :
  - Linux distribution Red Hat 4
  - Tomcat 5.5
  - Oracle 10g

[33] XHTML : eXtensible HyperText Mark-up Language

[34] HTML : HyperText Mark-up Language

- une architecture « poste développeur » :
  - Eclipse + plugin (3.1.2)
  - Log4J 1.2.13
  - Junit 3.8
  - Java JDK 1.5
- un langage de développement :
  - JAVA JDK version 1.5

Cette solution a été décrite dans le CDCT et donc validée par la MOA. Cependant, il est difficile pour le demandeur de réaliser la pertinence et la satisfaction de cette solution tant que le développement de l'application n'est pas mis en œuvre.

La réalisation des maquettes d'écran permet à la MOA d'avoir une idée plus précise de ce que pourrait être la solution finale : l'application MORSE.

3.3. Le développement

La phase de développement dans un projet informatique est l'écriture des programmes permettant d'obtenir l'application qui automatisera le ou les processus manuellement effectués.

Cette phase recouvre également les tests et la maintenance corrective en cas d'anomalies constatées à la mise en production.

Pour le projet MORSE, la phase de développement est la phase principale du cycle de vie d'un projet qui sera détaillée dans ce mémoire.

Après, une description de la programmation en JAVA de l'application, une description de la base de données Oracle 10g sera faite puis une description d'une connexion RPC et de Hibernate complèteront ce chapitre.

3.3.1. Utilisation d'un AGL : Eclipse

3.3.1.1. Définition d'un AGL

Les qualités d'un logiciel se mesurent selon plusieurs critères, mais elles concernent toutes le domaine de l'utilisation (fiabilité, ergonomie, adéquation aux besoins attendus, ...) et le domaine de la maintenance (correction, amélioration, documentation, ...).

Ainsi le génie logiciel est « l'art » de produire de bons logiciels avec un rapport qualité/prix optimal. Il fait appel à des principes d'ingénierie et au delà de la technique de programmation, à des méthodologies et des concepts informels (capacités d'interprétation, d'anticipation et de communication).
L'informaticien se trouve donc confronté au « savoir » (formation aux différentes techniques) et au « savoir-faire » (l'expérience).

Le génie logiciel a ainsi donné naissance aux Ateliers de Génie Logiciel (AGL) ou outils CASE[35] qui sont des logiciels d'aide à la réalisation de logiciels (système de développement logiciel assisté par ordinateur).
Un AGL intègre très souvent un ensemble d'outils qui permettent de réaliser les différentes phases d'un cycle de vie d'un logiciel. Cependant certains formalismes et méthodologies ne peuvent être entièrement réalisés sous AGL, il appartient à l'informaticien d'entreprendre ces tâches (concepts informels).

L'AGL doit permettre d'améliorer la qualité par le suivi des différentes phases processus et fournir un cadre cohérent et uniforme du produit.

Les outils CASE d'un AGL sont de deux types :

Les outils horizontaux	Les outils verticaux
Éditeur de texte gestion de projet gestion du dictionnaire de données administration et droits d'accès gestion de configuration documentation service de communication	Spécifications conception génération de codes IDE (Integrated Development Environnement) compilateurs génération d'interfaces homme-machine génération de tests validation prototypage maintenance

Des fonctions supplémentaires sont parfois disponibles tel que le « reverse engineering » (ou rétro-conception, action qui consiste à analyser un logiciel pour en

[35] CASE : Computer Aided Software Engineering

déterminer son fonctionnement, son code) ou la ré-utilisation des bibliothèques de composants.

Les AGL sont dits « upper case » ou « lower case ». Le premier terme définit un environnement de conception tandis que le deuxième définit un environnement de développement.

Environnement de conception « upper case »	Environnement de développement « lower case »
Outils d'édition des diagrammes dictionnaire de données édition de rapports générateurs de squelettes de code outils pour prototypage	PHASE IMPLEMENTATION ET TESTS éditeurs générateurs d'interfaces homme-machine SGBD (Système de Gestion de Base de Données) compilateurs optimiseurs debuggers ENVIRONNEMENT DEDIE bibliothèques de composants interface graphique éditeurs dédiés au langage interpréteurs debuggers GENERATEURS D'ENVIRONNEMENT éditeurs dédiés au langage pretty-printer (formatage automatique du code) debuggers interpréteurs

3.3.1.2. Eclipse, l'AGL du projet MORSE

Eclipse est un AGL en environnement de développement intégré (IDE en anglais). C'est un logiciel libre développé par Eclipse Foundation. Il repose sur un principe de plug-ins (extensions d'un logiciel principal), ainsi toutes les fonctionnalités d'Eclipse sont développées sous forme de plug-ins selon la norme OSGi (Open Services Gateway initiative).

Eclipse est écrit en JAVA avec l'aide des bibliothèques SWT[36] d'IBM, il permet de créer des projets de développement indépendamment du langage utilisé.

L'Eclipse Plateform qui est la base est composée de :

- *Plateform Runtime*, démarre la plateforme et gère les plug-ins
- *SWT*, la bibliothèque graphique de base
- *Jface*, une bibliothèque graphique plus perfectionnée, basée sur SWT
- *Eclipse Workbench*, la dernière couche graphique qui permet de manipuler des vues, des éditeurs et des perspectives.

Le projet *Eclipse RCP* (Rich Client Platform) offre la possibilité de développer des clients lourds indépendants d'Eclipse tout en réutilisant les composants de base.

Pour le projet MORSE, le choix de l'utilisation d'Eclipse s'est fait à partir du palier technique de la DGDDI. Ainsi le CDCT décrit une version 3.1.2 d'Eclipse. L'installation de cette version n'a pas pu être mise en œuvre car les exécutables d'installations n'étaient plus disponibles sur le site d'Eclipse. Ainsi, la première version installée sur mon poste de travail a été la version 3.2 (Yoxos). Les plug-ins pour l'utilisation de GWT disponibles sur les sites internet n'étaient pas compatibles avec cette version d'Eclipse. J'ai donc finalement installé la version 3.5 (Galileo) et les plug-ins nécessaire au fonctionnement de la plateforme de développement.

Le dernier palier technique version 2010C[37] de la DGDDI en date du mois de juin 2010 préconise la version 3.4 d'Eclipse (Ganymede).

Ainsi sur le poste développeur, je n'ai pas respecté le palier technique tout en respectant les autres préconisations de développement (JAVA, Oracle, ….).

[36] SWT : Standard Widget Toolkit

[37] Voir palier technique 2010C en annexe 17

3.3.1.3. Les avantages et les inconvénients de l'AGL pour le projet MORSE

Les inconvénients

Le principal inconvénient qui s'est imposé sur MORSE a été la recherche d'information. Eclipse étant un logiciel libre, même si le site officiel dispose de documentation, la plupart des informations sont données par des informaticiens qui exposent leur expérience et savoir-faire.

A travers cette multitude d'informations, il est nécessaire d'effectuer un premier tri pour essayer de déterminer quels sont celles qui seraient obsolètes.

L'utilisation de tutoriels peut s'avérer fastidieuse car ils doivent être déroulés en totalité pour parfois s'apercevoir qu'ils ne correspondent pas aux besoins du projet.

S'auto-former est ainsi consommateur de temps et a rallongé les délais de développement prévus.

A travers le projet MORSE, cet état de fait a été clairement identifié puisque j'ai dû faire appel à un programmeur expérimenté pour me donner une formation “accélérée” sur la connexion et l'utilisation des bases de données de l'application.

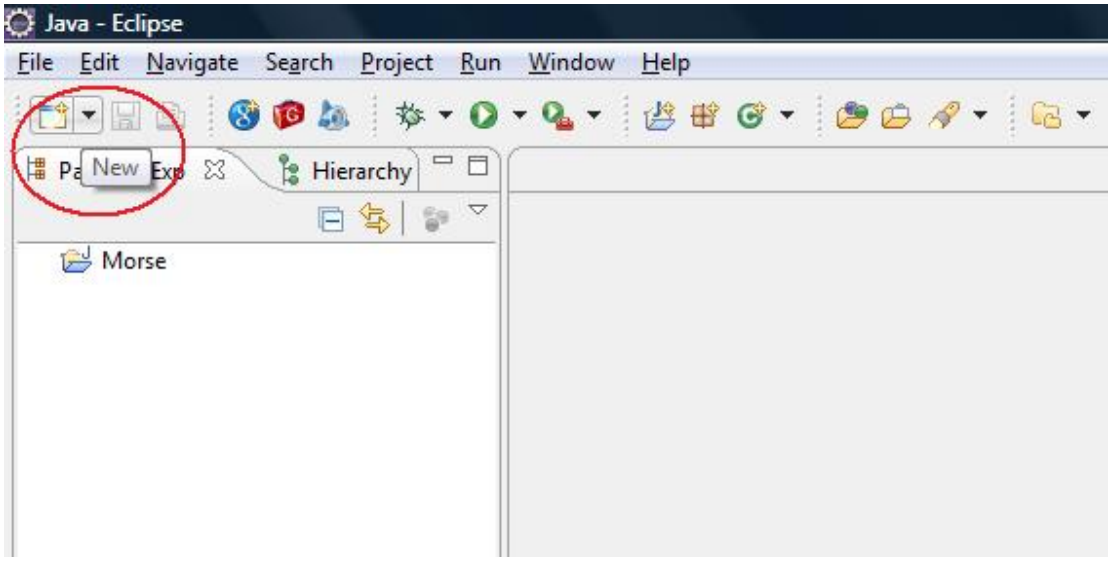
L'AGL Eclipse ne peut être destiné à des novices en matière de programmation, il est nécessaire d'avoir une solide base en programmation objet ou d'avoir déjà programmé en un autre langage pour prétendre s'auto-former.

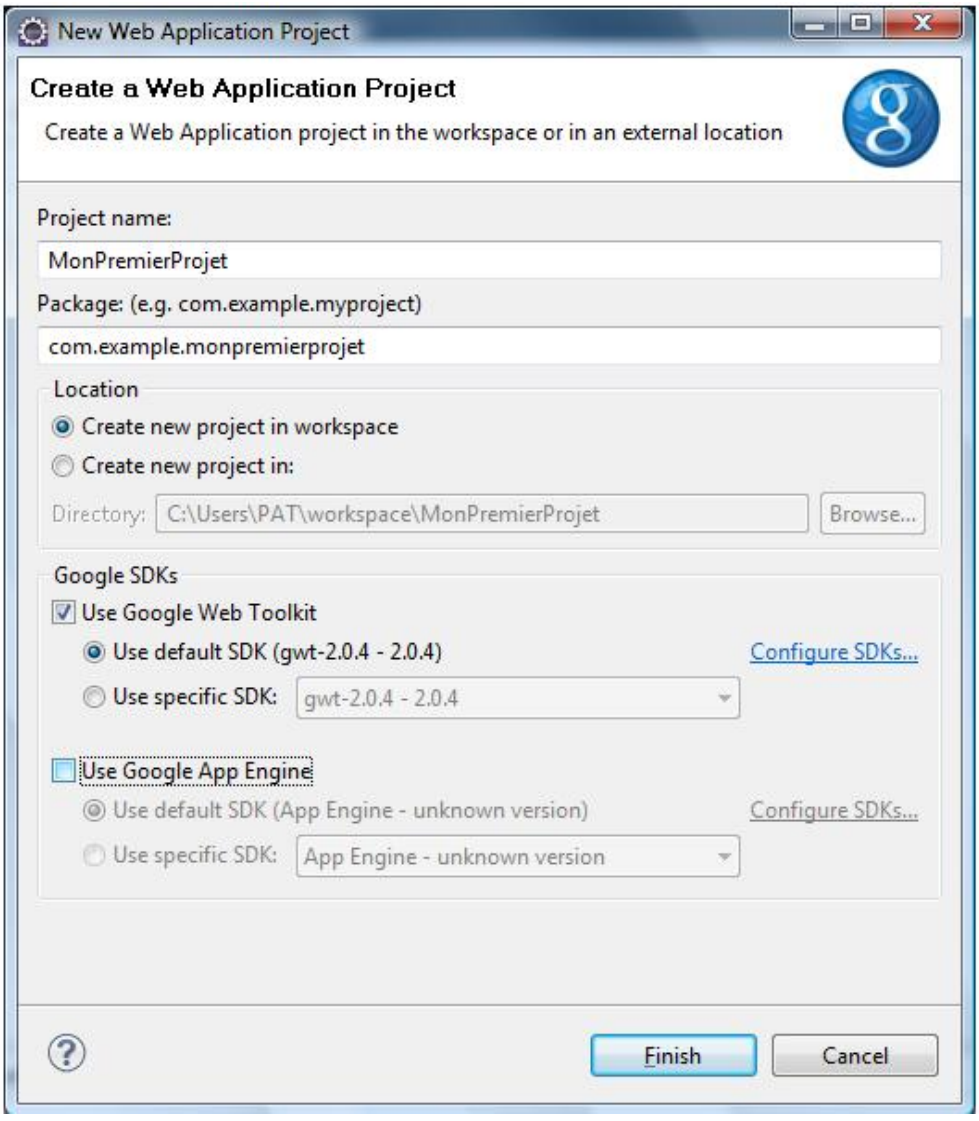
Les avantages

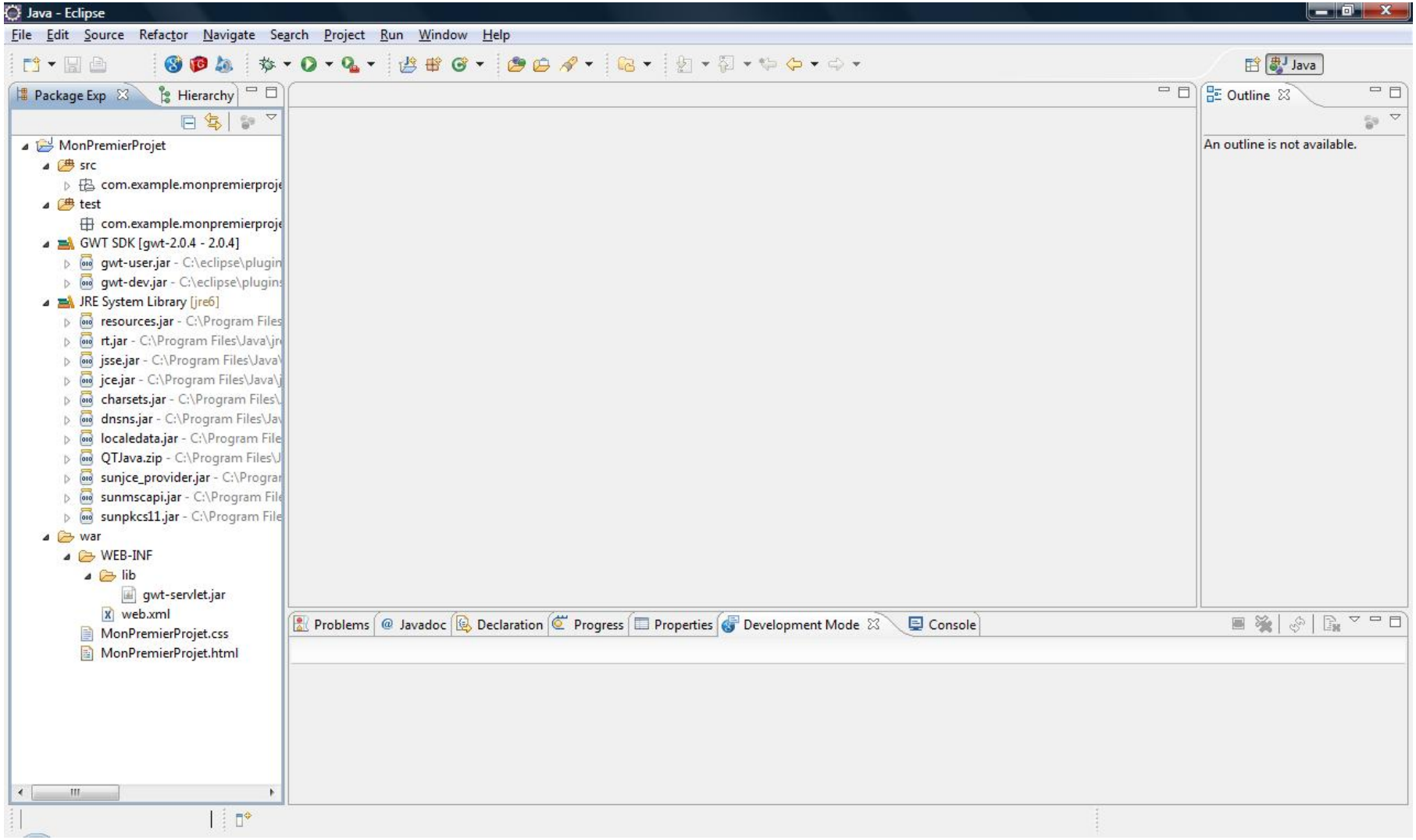
Une fois, les fonctionnalités de bases d'Eclipse maîtrisées, cet AGL s'avère puissant et économiseur de temps.

Création rapide d'une application

Il permet en quelques clics de créer la trame des applications avec une arborescence de fichiers comprenant toutes les classes et interfaces initiales.


Java - Eclipse
File Edit Navigate Search Project Run Window Help
New
Hierarchy
Morse


New Web Application Project
Create a Web Application Project
Create a Web Application project in the workspace or in an external location
Project name:
MonPremierProjet
Package: (e.g. com.example.myproject)
com.example.monpremierprojet
Location
Create new project in workspace
Create new project in:
Directory:
C:\Users\PAT\workspace\MonPremierProjet
Browse...
Google SDKs
Use Google Web Toolkit
Use default SDK (gwt-2.0.4 - 2.0.4)
Configure SDKs...
Use specific SDK:
gwt-2.0.4 - 2.0.4
Use Google App Engine
Use default SDK (App Engine - unknown version)
Configure SDKs...
Use specific SDK:
App Engine - unknown version
Finish
Cancel


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package Exp
Hierarchy
MonPremierProjet
src
com.example.monpremierproje
test
com.example.monpremierproje
GWT SDK [gwt-2.0.4 - 2.0.4]
gwt-user.jar - C:\eclipse\plugin
gwt-dev.jar - C:\eclipse\plugins
JRE System Library [jre6]
resources.jar - C:\Program Files
rt.jar - C:\Program Files\Java\jr
jsse.jar - C:\Program Files\Java\
jce.jar - C:\Program Files\Java\j
charsets.jar - C:\Program Files\
dnsns.jar - C:\Program Files\Jav
localedata.jar - C:\Program File
QTJava.zip - C:\Program Files\J
sunjce_provider.jar - C:\Progra
sunmscapi.jar - C:\Program File
sunpkcs11.jar - C:\Program File
war
WEB-INF
lib
gwt-servlet.jar
web.xml
MonPremierProjet.css
MonPremierProjet.html
Outline
An outline is not available.
Problems
Javadoc
Declaration
Progress
Properties
Development Mode
Console

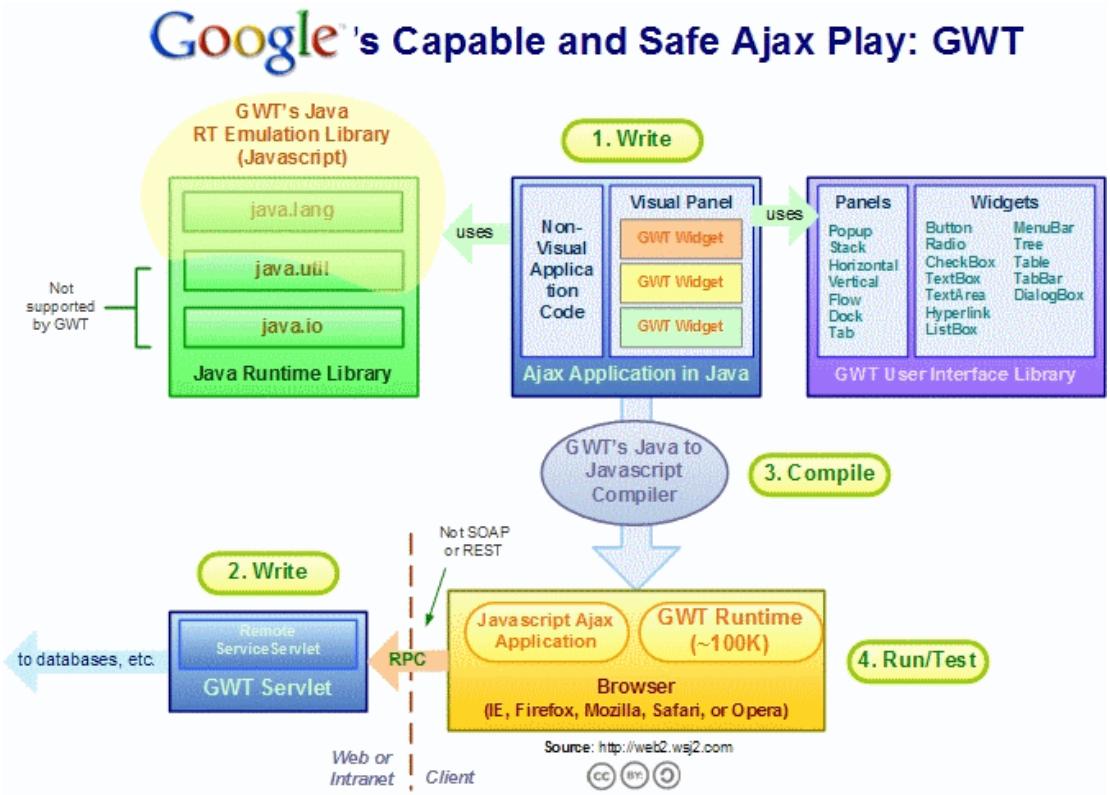
Utilisation de widgets

De plus l'utilisation de GWT donne la possibilité de récupérer des codages d'objets graphiques et ainsi de ne pas avoir à les coder.

GWT est un framework qui facilite le développement d'application web. Il a été développé parce que le javascript n'était pas agréable à écrire, difficile à débugger, les environnements de développement plus adaptés ou inexistants. Ainsi, seules des compétences en JAVA sont nécessaires pour utiliser GWT, le fonctionnement interne étant pris en charge.

L'architecture globale du framework :

- les widgets "GWT" qui permettent d'obtenir une interface graphique
- le compilateur "GWT" propriétaire qui transforme le code Java en "Javascript"
- l'émulation des librairies Java par le compilateur pour les transformer en Javascript
- l'utilisation du RPC pour l'échange "Client/Serveur"



Il reste donc au développeur à :

- Ecrire l'interface graphique, en utilisant les widgets fournis par GWT, et les émulations de librairies Java
- Ecrire les servlets si besoin
- Compiler le code avec le compilateur "Java-To-Javascript"
- Tester/exécuter l'application

Dans l'application MORSE, un bouton graphique se «dessine» avec GWT de la façon suivante :

- on commence par importer les librairies adéquates

```
import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.VerticalPanel;
```

- puis on définit une page, un panneau vertical et un bouton, on les crée et les formate

```
private PageMatricule pm;
VerticalPanel vPanel = new VerticalPanel();
vPanel.setSpacing(5);
vPanel.addStyleName("verticalPanel");
private Button sendButton = new Button("Gestion Agent");
sendButton.addStyleName("sendButton");
```

- on ajoute ce bouton au panneau vertical

```
vPanel.add(sendButton);
```

- on définit une action sur ce bouton

```
// click sur bouton Agent
    sendButton.addClickHandler(new ClickHandler() {
    public void onClick(ClickEvent event) {
    RootPanel.get().remove(0);
    pm = new PageMatricule();
    RootPanel.get().add(pm);
    }
```

●on affiche la page

```
initWidget(vPanel);
```

●On obtient le résultat suivant :

Normalisation de l'application

Eclipse permet de normaliser la programmation en permettant :

●l'identation automatique par la fonctionnalité «Source/Correct identation»,

●la génération automatique de commentaires

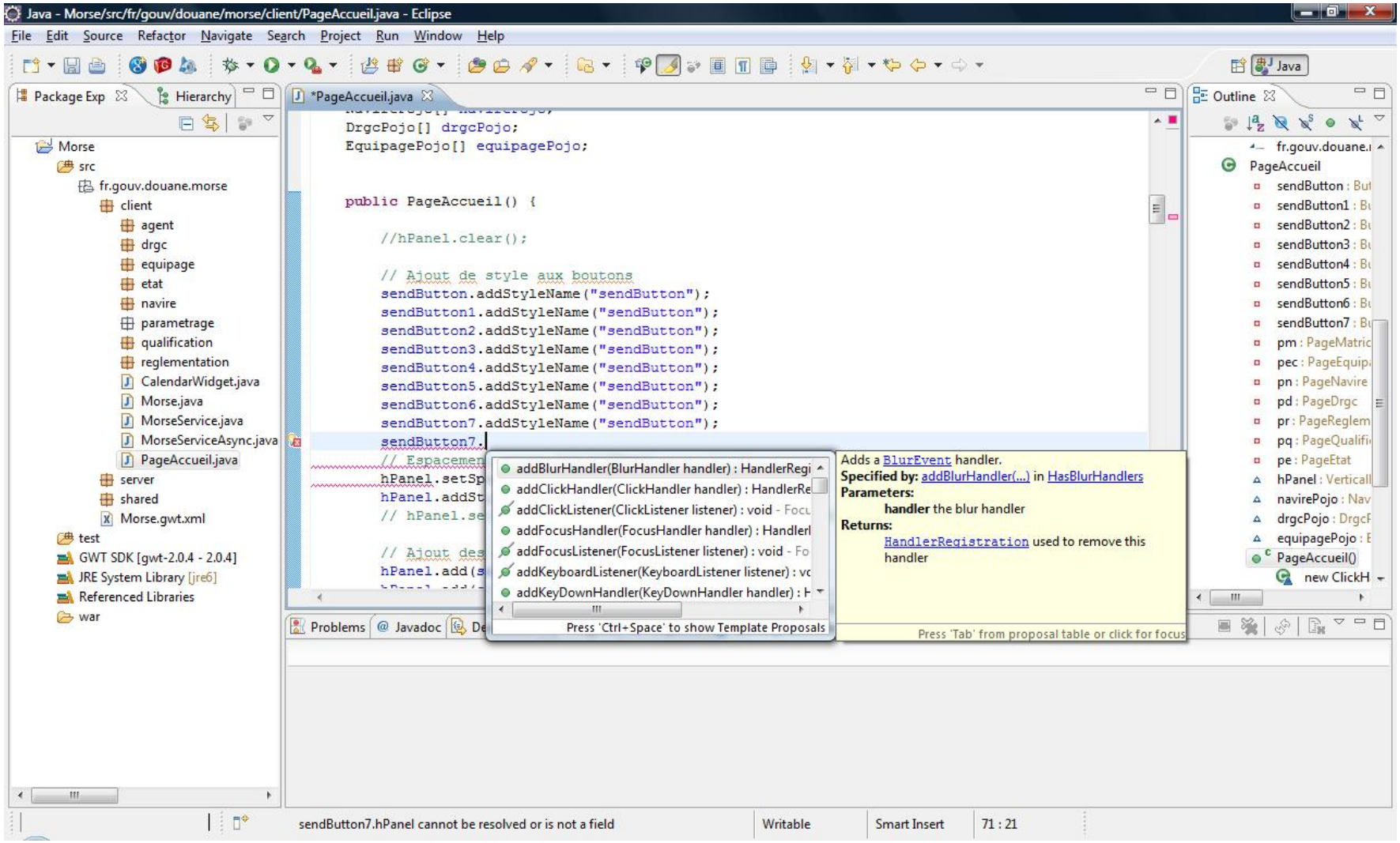
il suffit de taper «`/** envoi du matricule au serveur` et faire <Entrer> pour obtenir

```
/** envoi du matricule au serveur
*
* @param matricule
*/
```

Aide à la programmation

Eclipse permet au programmeur de disposer de commandes «suggérées» qu'il lui suffit de cliquer pour mettre en œuvre.

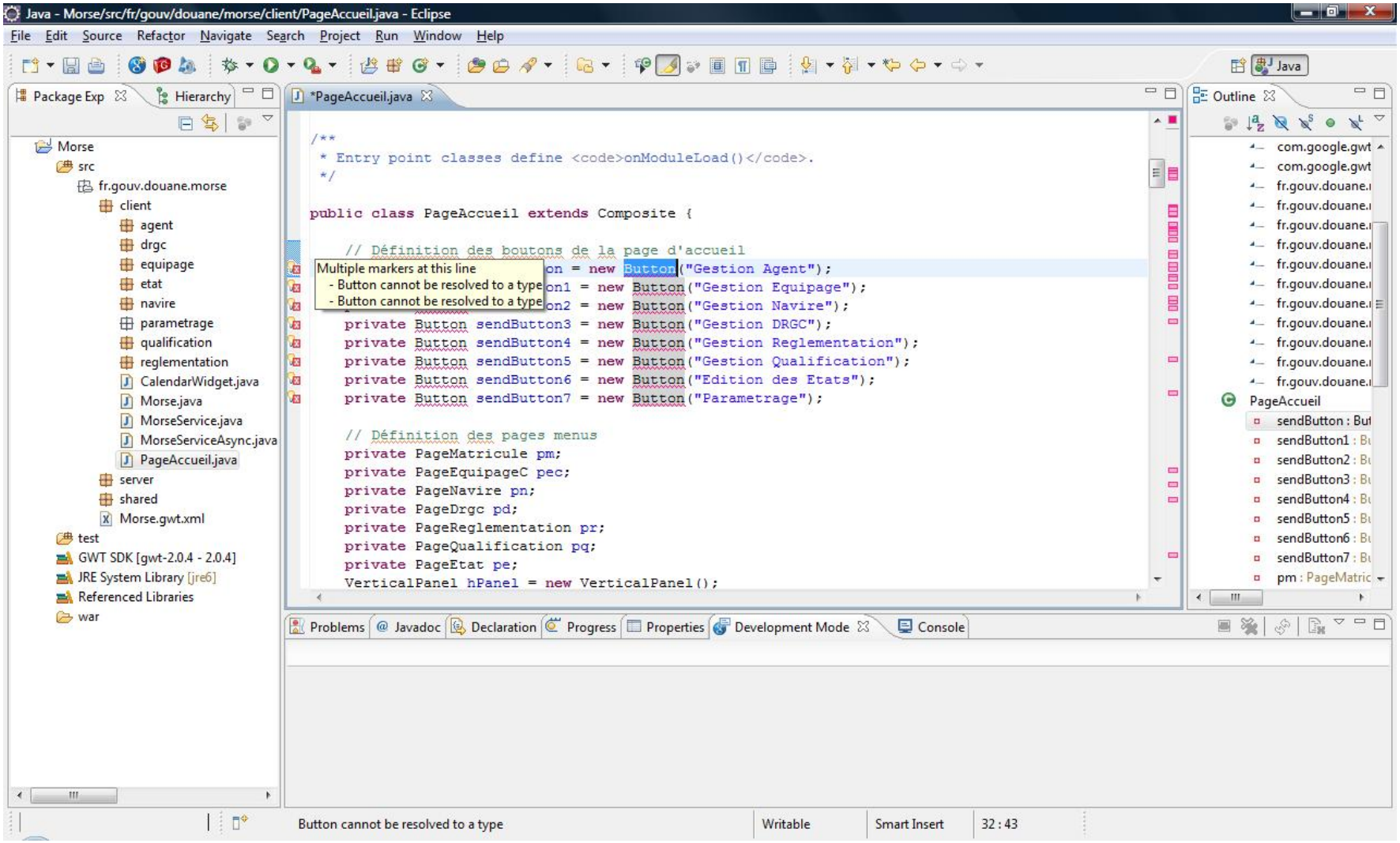
Cette fonctionnalité de l'AGL constitue un gain de temps non négligeable.

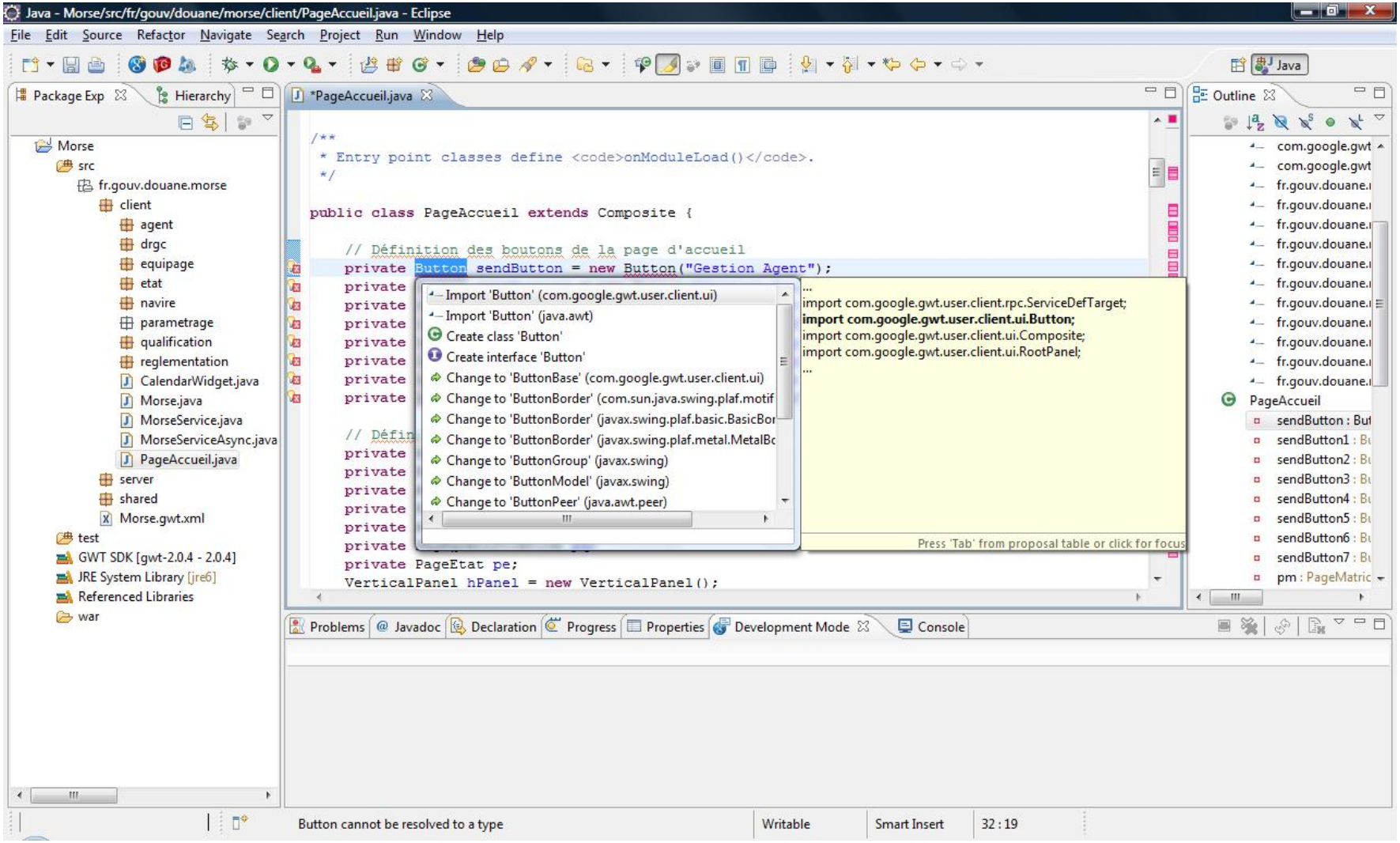

Java - Morse/src/fr/gouv/douane/morse/client/PageAccueil.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Exp
Hierarchy
*PageAccueil.java
Outline
Morse
src
fr.gouv.douane.morse
client
agent
drgc
equipage
etat
navire
parametrage
qualification
reglementation
CalendarWidget.java
Morse.java
MorseService.java
MorseServiceAsync.java
PageAccueil.java
server
shared
Morse.gwt.xml
test
GWT SDK [gwt-2.0.4 - 2.0.4]
JRE System Library [jre6]
Referenced Libraries
war
DrgcPojo[] drgcPojo;
EquipagePojo[] equipagePojo;
public PageAccueil() {
//hPanel.clear();
// Ajout de style aux boutons
sendButton.addStyleName("sendButton");
sendButton1.addStyleName("sendButton");
sendButton2.addStyleName("sendButton");
sendButton3.addStyleName("sendButton");
sendButton4.addStyleName("sendButton");
sendButton5.addStyleName("sendButton");
sendButton6.addStyleName("sendButton");
sendButton7.addStyleName("sendButton");
sendButton7.
addBlurHandler(BlurHandler handler) : HandlerRegi
addClickHandler(ClickHandler handler) : HandlerRe
addClickListener(ClickListener listener) : void - Focu
addFocusHandler(FocusHandler handler) : HandlerI
addFocusListener(FocusListener listener) : void - Fo
addKeyboardListener(KeyboardListener listener) : vo
addKeyDownHandler(KeyDownHandler handler) : H
Press 'Ctrl+Space' to show Template Proposals
Adds a BlurEvent handler.
Specified by: addBlurHandler(...) in HasBlurHandlers
Parameters:
handler the blur handler
Returns:
HandlerRegistration used to remove this handler
Press 'Tab' from proposal table or click for focus
Problems
Javadoc
fr.gouv.douane.
PageAccueil
sendButton : But
sendButton1 : Bu
sendButton2 : Bu
sendButton3 : Bu
sendButton4 : Bu
sendButton5 : Bu
sendButton6 : Bu
sendButton7 : Bu
pm : PageMatric
pec : PageEquip
pn : PageNavire
pd : PageDrgc
pr : PageReglem
pq : PageQualifi
pe : PageEtat
hPanel : VerticalI
navirePojo : Nav
drgcPojo : DrgcP
equipagePojo : E
PageAccueil()
new ClickH
sendButton7.hPanel cannot be resolved or is not a field
Writable
Smart Insert
71 : 21

Débogage performant

Eclipse permet de débuguer une application facilement grâce à des messages explicites et à des propositions de correction.

Visuellement le code en erreur est signalé par des pictogrammes qui alertent le programmeur.


Java - Morse/src/fr/gouv/douane/morse/client/PageAccueil.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Exp
Hierarchy
*PageAccueil.java
Outline
Morse
src
fr.gouv.douane.morse
client
agent
drgc
equipage
etat
navire
parametrage
qualification
reglementation
CalendarWidget.java
Morse.java
MorseService.java
MorseServiceAsync.java
PageAccueil.java
server
shared
Morse.gwt.xml
test
GWT SDK [gwt-2.0.4 - 2.0.4]
JRE System Library [jre6]
Referenced Libraries
war
/**
* Entry point classes define <code>onModuleLoad()</code>.
*/
public class PageAccueil extends Composite {
// Définition des boutons de la page d'accueil
Multiple markers at this line
- Button cannot be resolved to a type
- Button cannot be resolved to a type
on = new Button("Gestion Agent");
on1 = new Button("Gestion Equipage");
on2 = new Button("Gestion Navire");
private Button sendButton3 = new Button("Gestion DRGC");
private Button sendButton4 = new Button("Gestion Reglementation");
private Button sendButton5 = new Button("Gestion Qualification");
private Button sendButton6 = new Button("Edition des Etats");
private Button sendButton7 = new Button("Parametrage");
// Définition des pages menus
private PageMatricule pm;
private PageEquipageC pec;
private PageNavire pn;
private PageDrgc pd;
private PageReglementation pr;
private PageQualification pq;
private PageEtat pe;
VerticalPanel hPanel = new VerticalPanel();
com.google.gwt
com.google.gwt
fr.gouv.douane.
PageAccueil
sendButton : But
sendButton1 : Bu
sendButton2 : Bu
sendButton3 : Bu
sendButton4 : Bu
sendButton5 : Bu
sendButton6 : Bu
sendButton7 : Bu
pm : PageMatric
Problems
Javadoc
Declaration
Progress
Properties
Development Mode
Console
Button cannot be resolved to a type
Writable
Smart Insert
32 : 43


Java - Morse/src/fr/gouv/douane/morse/client/PageAccueil.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Exp
Hierarchy
*PageAccueil.java
Outline
Morse
src
fr.gouv.douane.morse
client
agent
drgc
equipage
etat
navire
parametrage
qualification
reglementation
CalendarWidget.java
Morse.java
MorseService.java
MorseServiceAsync.java
PageAccueil.java
server
shared
Morse.gwt.xml
test
GWT SDK [gwt-2.0.4 - 2.0.4]
JRE System Library [jre6]
Referenced Libraries
war
/**
* Entry point classes define <code>onModuleLoad()</code>.
*/
public class PageAccueil extends Composite {
// Définition des boutons de la page d'accueil
private Button sendButton = new Button("Gestion Agent");
private PageEtat pe;
VerticalPanel hPanel = new VerticalPanel();
Import 'Button' (com.google.gwt.user.client.ui)
Import 'Button' (java.awt)
Create class 'Button'
Create interface 'Button'
Change to 'ButtonBase' (com.google.gwt.user.client.ui)
Change to 'ButtonBorder' (com.sun.java.swing.plaf.motif
Change to 'ButtonBorder' (javax.swing.plaf.basic.BasicBor
Change to 'ButtonBorder' (javax.swing.plaf.metal.MetalBc
Change to 'ButtonGroup' (javax.swing)
Change to 'ButtonModel' (javax.swing)
Change to 'ButtonPeer' (java.awt.peer)
...
import com.google.gwt.user.client.rpc.ServiceDefTarget;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.RootPanel;
...
Press 'Tab' from proposal table or click for focus
PageAccueil
sendButton : Buf
sendButton1 : Bu
sendButton2 : Bu
sendButton3 : Bu
sendButton4 : Bu
sendButton5 : Bu
sendButton6 : Bu
sendButton7 : Bu
pm : PageMatric
Problems
Javadoc
Declaration
Progress
Properties
Development Mode
Console
Button cannot be resolved to a type
Writable
Smart Insert
32 : 19

Les avantages énumérés ci-dessus sont ceux dont j'ai bénéficié pour la programmation de l'application MORSE.

Cependant bien d'autres fonctionnalités sont disponibles et facilitent d'autant le développement d'une application web :

- ●production de la documentation JAVA
- ●génération de tests unitaires,
- ●refactoring, fonctionnalité qui permet de modifier en cascade. Si on renomme une classe par exemple, Eclipse se chargera d'effectuer toutes les modifications nécessaires dans l'application développée.
- ●Correction orthographique
- ●nettoyage automatique de code

3.3.2. la programmation en JAVA de l'application MORSE

Afin de respecter le palier technique imposé par la DGDDI, la programmation de Morse a été effectuée en langage JAVA.

La finalité de l'application MORSE est de gérer le suivi des personnels spécialisés et la composition des équipages de la DRGC-MMN et de produire des états prévisionnels pour anticiper les échéances de revalidation de qualifications et d'aptitudes médicales.

Ainsi l'application a été conçue afin de permettre la saisie des informations de base (agents, équipages, navires, DRGC, réglementations, qualifications) et de proposer des états prévisionnels.

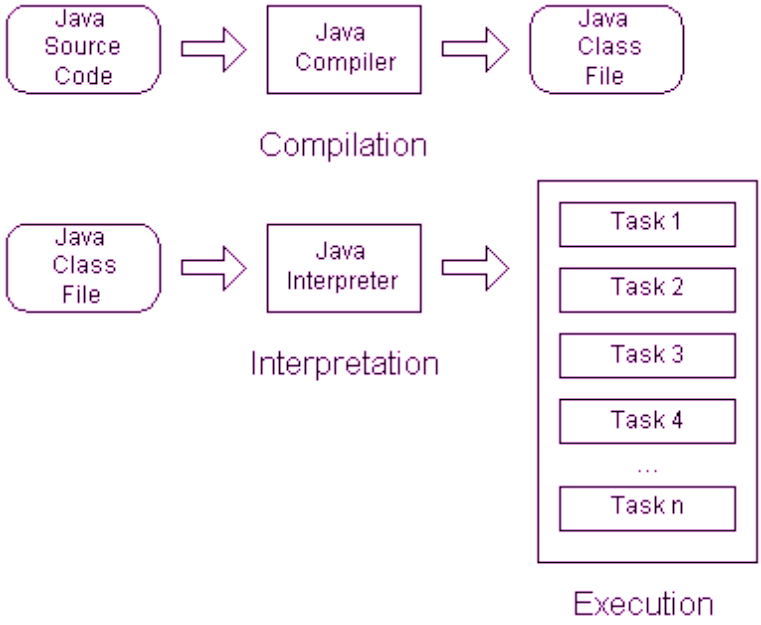
3.3.2.1. Le langage JAVA

JAVA est conçu pour être complètement orienté objet. Il est basé sur la première version du langage C++ épurée des concepts contraignants et particuliers des pointeurs et références. Il intègre les propriétés d'héritage et de polymorphisme et les expressions et structures de contrôle du langage C, abondé des exceptions. Même si un programme en JAVA est considéré comme un langage très verbeux, au final il apparait beaucoup plus compréhensible que le code généré sous C++.

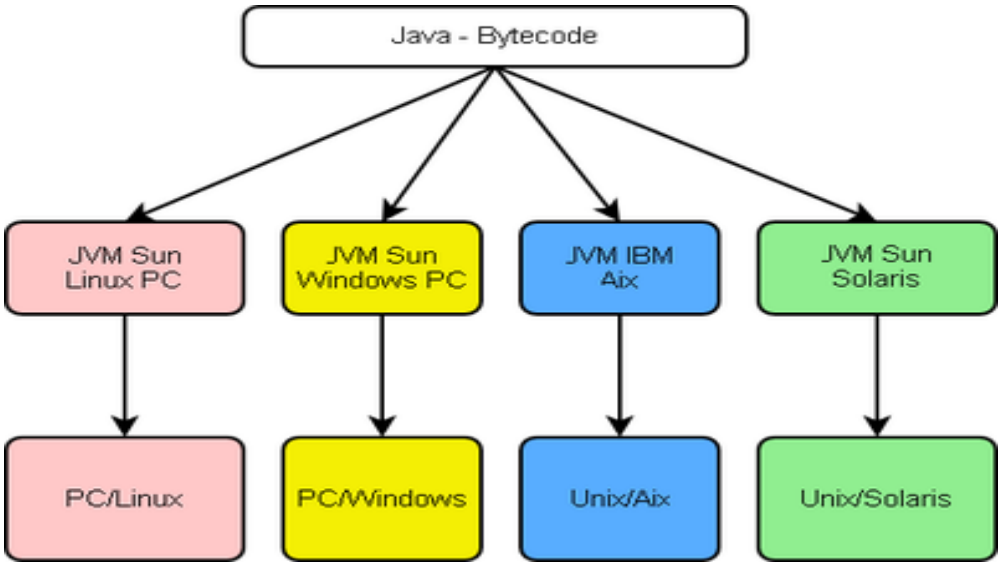
JAVA est compilé dans un langage machine spécial qui sera ensuite interprété, il est légèrement moins rapide à l'exécution que le C++. Cependant du code compilé en JAVA peut être exécuté sur différents environnements, on parle de « Compile-Once-Run-Anywhere ». Un programme JAVA peut ainsi être transmis par le réseau à une autre machine avec un système d'exploitation et des interfaces graphiques différents.

La première version de JAVA est JDK 1.0 (Java Development Kit) et a été éditée par Sun Microsystems.
La dernière version actuellement utilisée, JAVA SE6 (Java Standard Edition) a été éditée le 12 décembre 2006.
Une version JAVA SE7 est en cours de spécifications.
De la première version à la version actuelle, de nombreux changements au niveau du langage et l'ajout de nombreuses classes (d'une centaine dans la version d'origine à plus de 3 000) ont été faits.
En 2009, la société Oracle rachetait la société Sun Microsystems, d'où la présence du logo Oracle dans les documents de l'API[38] JAVA.
Le langage JAVA est utilisé pour écrire des applications, des applets et des servlets.
Une *application* JAVA est similaire à un programme écrit dans un langage de 3ème génération, est compilée et exécutée sur une même machine.
Le bytecode Java est le résultat de la compilation d'un programme dont le code source est en Java.

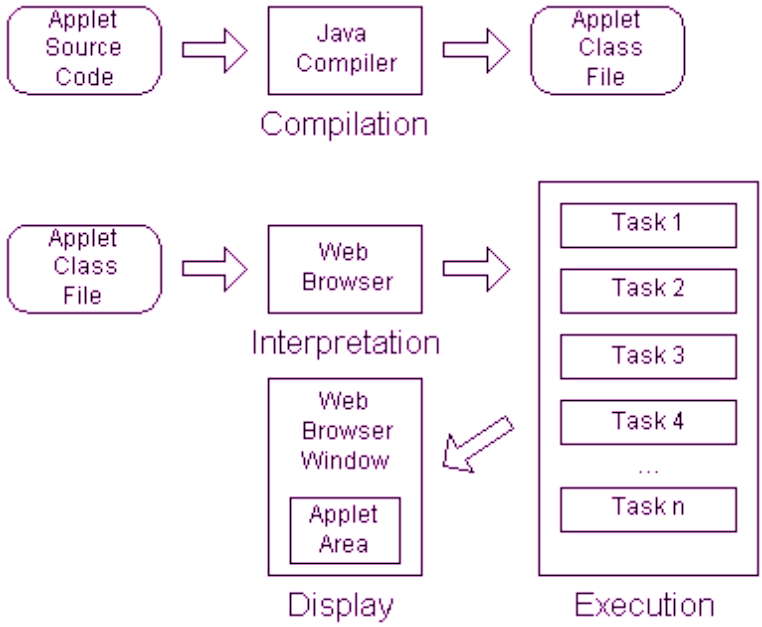
[38] API : Application Programming Interface



La Java virtual machine (JVM) est une machine virtuelle permettant d'interpréter et d'exécuter le bytecode Java. Ce programme est spécifique à chaque plate-forme ou couple machine/système d'exploitation et permet aux applications Java compilées en bytecode de produire les mêmes résultats quelle que soit la plate-forme, tant que celle-ci est pourvue de la machine virtuelle Java adéquate. La machine virtuelle la plus utilisée est celle de Sun Microsystems.



Un *applet* JAVA est compilé sur une machine, stocké en binaire sur un serveur et peut être envoyé à travers Internet pour être interprété par un navigateur supportant JAVA.

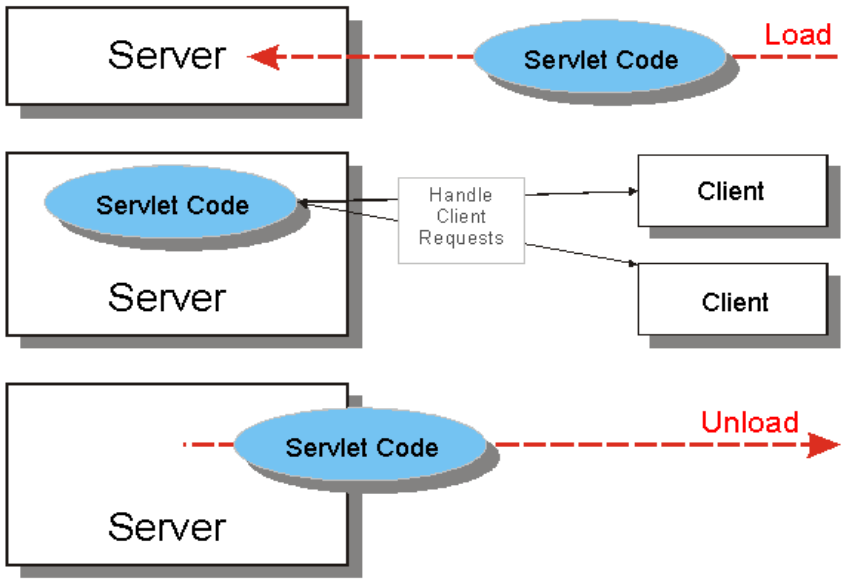


Un *servlet* JAVA permet de créer dynamiquement des données en format HTTP[39], il s'exécute sur le serveur web en étendant ses fonctions (e-commerce, accès à une base de données). Les servlets s'exécutent indépendamment de la plateforme utilisée et sans limitation de performance imposée par les programmes CGI (Common Gateway Interface). Ils se chargent automatiquement au démarrage de l'application ou à la première requête client et restent disponibles pour les requêtes suivantes, diminuant ainsi la charge sur le serveur web.

Les servlets peuvent accéder à toutes les API[40] JAVA y compris les API JDBC pour accéder aux bases de données. Ils ont également accès aux bibliothèques HTTP et héritent de tous les bénéfices du langage JAVA : portabilité, performance, ré-utilisation.

[39] HTTP : HyperText Transfert Protocol

[40] API : Application Programming Interface


Server
Servlet Code
Load
Servlet Code
Handle
Client
Requests
Client
Client
Server
Servlet Code
Unload
Server

Le langage JAVA dispose d'une grande bibliothèque de classes et d'objets prêts à être utilisés.

PACKAGES	DESCRIPTION
java.lang	*Classes de base.* comprend les classes de base de Java qui sont automatiquement importées par le compilateur.
java.util	*Gestion de données et utilitaires.* rassemble des classes d'utilitaires (gestion des collections de données, génération de nombres aléatoires, énumération, date,...). Il définit aussi les classes d'exceptions (EmptyStackException, NoSuchElementException).
java.io	*Les entrées-sorties.* Ce package regroupe les classes permettant de gérer les entrées-sorties (accès fichiers, gestion de répertoires,...). Il définit aussi les classes d'exceptions IOException, EOFException, FileNotFoundException, InterruptedIOException, UTFDataFormatException.
java.net	*Les accès réseau.* comprend les classes permettant de gérer les accès réseau. Il définit aussi les classes d'exceptions MalformedURLException, ProtocolException, SocketException, UnknownHostException, UnknownServiceException.
java.applet	*Gestion des applets.* La classe Applet et les interfaces de ce package permettent de programmer une applet Java et d'intégrer une applet dans un navigateur.
java.awt	*Interface utilisateur.* Ces classes permettent la programmation de l'interface graphique d'un programme ou d'une applet Java. Il définit aussi les classes d'exceptions AWTException et AWTError.
java.awt.image	*Manipulation d'images.* Ce package permet de manipuler les images (chargement des images, filtres, gestion des couleurs, ...)
java.awt.peer	*Liaison avec l'interface utilisateur.* Ensemble d'interfaces qui permettent la représentation à l'écran des composants graphiques JAVA (boutons, fenêtres, ...) sur un système qui supporte une Machine Virtuelle Java (MVJ).

L'utilisation de JAVA dans des domaines très diversifiés se fait grâce à la mise à disposition de « frameworks » et d'API.

Un « framework » est un ensemble de composants logiciels structurels et d'outils qui définissent l'organisation de tout ou partie d'un logiciel.

Les quatre « frameworks » principaux fournis par Sun Microsystems sont :

- J2SE : pour les applications pour poste de travail, désormais dénommé JAVA SE
- J2EE : pour les applications serveurs. Ce framework contient un grand nombre d'API et d'extensions. Il est désormais appelé JAVA EE.
- J2ME : spécialement conçu pour les applications mobiles, il est dénommé JAVA ME.
- JAVACard : ce framework est réservé aux applications pour cartes à puces et SmartCards.

3.3.2.2. Les fonctionnalités de MORSE programmées en JAVA

Le Modèle-Vue-Contrôleur (MVC) est une architecture et une méthode de conception qui organise l'interface homme-machine (IHM) d'une application logicielle. Ce paradigme divise l'IHM en un modèle (modèle de données), une vue (présentation, interface utilisateur) et un contrôleur (logique de contrôle, gestion des événements, synchronisation), chacun ayant un rôle précis dans l'interface.

La partie « modèle » recouvre la base de données , les POJO[41], les DAO[42] et le manager situés du côté « serveur ».

Les POJO n'implémentent pas d'interface spécifique à un framework comme c'est le cas par exemple pour un composant EJB (Enterprise JavaBeans). Ils décrivent les propriétés des objets persistants issus de la base de données.

Les DAO regroupent un ensemble d'accès aux données persistantes, ils constituent des modèles de conception pour les architectures logicielles.

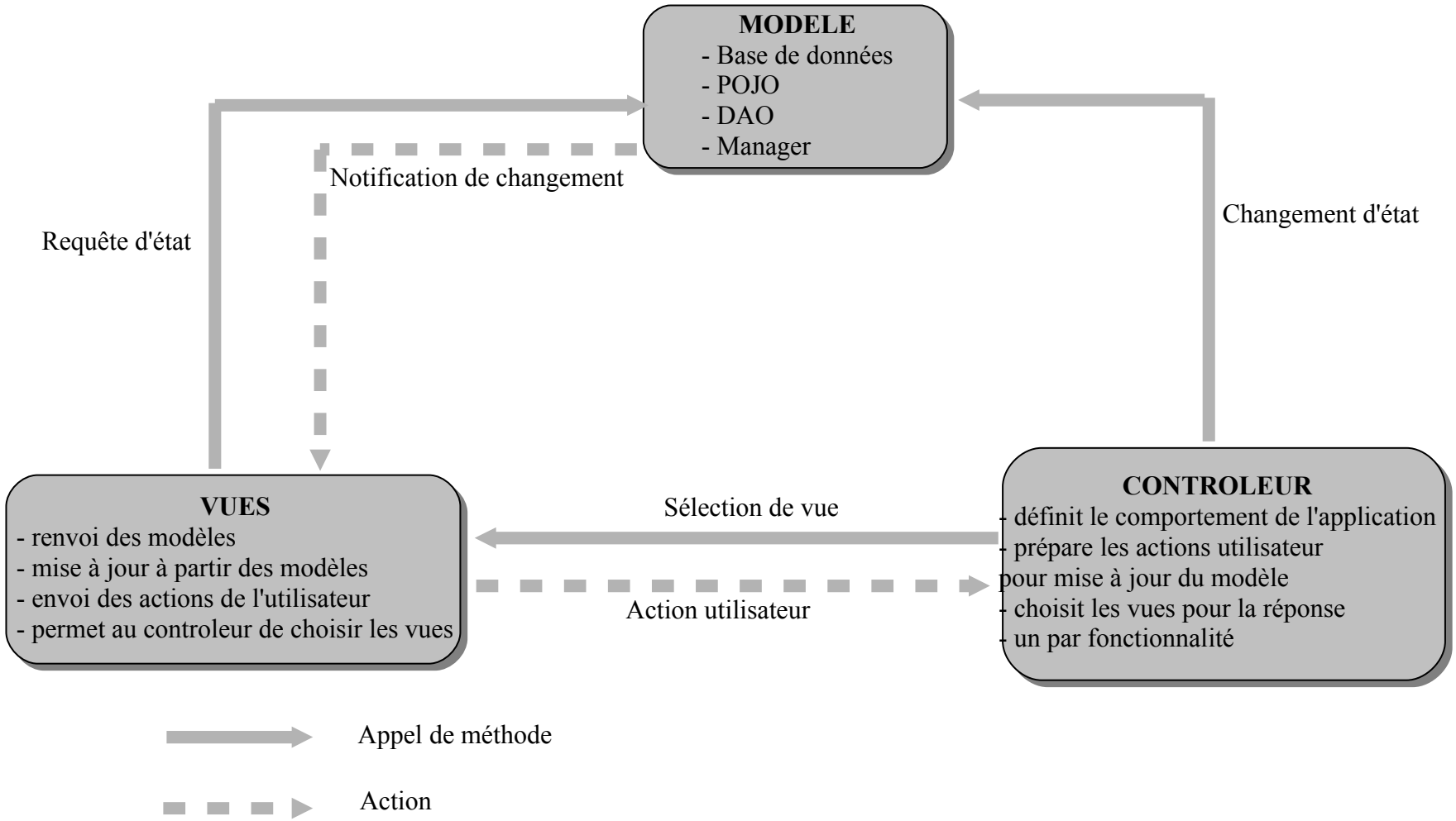
Le manager décrit les actions menées sur les données persistantes.

La partie « vues » est la présentation côté « client » des résultats obtenus.

La partie « contrôleur » comprend les services d'accès à la base de données côté « client ».

[41] POJO : Plain Old Java Object

[42] DAO : Data Access Object


MODELE
- Base de données
- POJO
- DAO
- Manager
Notification de changement
Requête d'état
Changement d'état
VUES
- renvoi des modèles
- mise à jour à partir des modèles
- envoi des actions de l'utilisateur
- permet au controleur de choisir les vues
Sélection de vue
Action utilisateur
CONTROLEUR
- définit le comportement de l'application
- prépare les actions utilisateur
pour mise à jour du modèle
- choisit les vues pour la réponse
- un par fonctionnalité
Appel de méthode
Action

L'application MORSE a été développée selon le schéma standard « modèle-vues-contrôleur» défini pour toute application web.

L'arborescence du projet MORSE se présente comme ci-dessous :

- ●morse
  - ○client
    - ▪agent
      - ◆*PageAgentC.java*
      - ◆*PageAgentM.java*
      - ◆*PageMatricule.java*
    - ▪drgc
      - ◆*PageDrgc.java*
    - ▪equipage
      - ◆*PageEquipageC.java*
    - ▪etat
      - ◆*PageEtat.java*
    - ▪navire
      - ◆*PageNavire.java*
    - ▪parametrage
      - ◆*PageParamétrage.java*
    - ▪qualification
      - ◆*PageQualification.java*
    - ▪reglementation
      - ◆*PageReglementation.java*
    - ▪*CalendarWidget.java*
    - ▪*Morse.java*
    - ▪*MorseService.java*
    - ▪*MorseServiceAsync.java*
    - ▪*PageAccueil.java*

- serveur
  - report
    - *reportAgent.jasper*
    - *reportAgent.jrxml*
    - *reportQualif.jasper*
    - *reportQualif.jrxml*
  - *DAO_Agent.java*
  - *DAO_Drgc.java*
  - *DAO_Equipage.java*
  - *DAO_Navire.java*
  - *DAO_Qualif.java*
  - *DB_Conn.java*
  - *GetReport.java*
  - *MorseServiceImpl.java*
- shared
  - *AgentPojo.java*
  - *DrgcPojo.java*
  - *EquipagePojo.java*
  - *FieldVerifier.java*
  - *NavirePojo.java*
  - *OraErr.java*
  - *QualifPojo.java*

La partie « modèle » reprend donc les répertoires « serveur » et « shared » qui comprennent les DAO et les POJO, ainsi que le fichier *MorseServiceImpl.java* (manager).

Le répertoire « report » quant à lui rassemble les états produits à l'aide de JasperReports[43].

[43] Logiciel de reporting gratuit de SourceForge

La partie « contrôleur » regroupe les fichiers *MorseService.java* et *MorseServiceAsync.java*

La partie « vues » comprend toutes les pages qui seront affichées à l'écran de l'utilisateur, elles se situent dans le répertoire « client ». Le fichier de la page d'accueil est également repris dans ce répertoire.

L'exemple qui suit, décrit la mise en œuvre des mécanismes d'appel entre les méthodes et les classes de l'application selon le schéma « modèle, contrôleur, vues ».

La partie « modèle ».

Le fichier *DrgcPojo.java* décrit l'objet POJO et les accesseurs et mutateurs qui lui sont rattachés.

Un accesseur est une méthode qui permet de récupérer le contenu d'une donnée membre protégée alors qu'un mutateur en permet la modification. Ils sont respectivement identifiés par « get » et « set ».

```
package fr.gouv.douane.morse.shared;
import com.google.gwt.user.client.rpc.IsSerializable;
public class DrgcPojo implements IsSerializable {
    private Integer id;
    private String nom;

    public DrgcPojo() {
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }

    public String getNom() {
        return nom;
    }
    public void setNom(String nom) {
        this.nom = nom;
    }
}
```

Les données de la table T_DRGC sont au nombre de deux (id et nom) et on distingue les méthodes « get » et « set » pour chacune de ces données.

Le fichier *DAO_Drgc.java* décrit les actions de recherche, de suppression et de modification entreprises sur les données de la base. Pour les accomplir le fichier comprend les connexions/déconnexions à la base de données, les requêtes et leurs exécutions et le résultat obtenu. Les classes de la partie « vues » font appel à ce fichier pour récupérer les données à traiter.

```
package fr.gouv.douane.morse.server;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import oracle.jdbc.rowset.OracleCachedRowSet;
import fr.gouv.douane.morse.shared.DrgcPojo;

/** étend la classe abstraite DB_Conn */

public class DAO_Drgc extends DB_Conn {

    public DAO_Drgc() {
        // null
    }

    /** Recherche des DRGC */

    public DrgcPojo[] getAllDrgc() {
        String query = "Select ID, NOM From T_DRGC order by NOM";
        // préparation pour le transport rpc
        DrgcPojo[] drgcPojo = null;

        try {
                Connection connection = getConn();
                PreparedStatement pstmt = connection.prepareStatement(query);
                ResultSet rst = pstmt.executeQuery();
                // Affecte le resultset dans le Pojo
```

```
                    drgcPojo = populate(rst);
                    rst.close();
                    connection.close();
                    }


                catch (Exception e) {
                    System.err.println("SQL Statement Error: " + query);
                    e.printStackTrace();
                    }

                // return the array
                return drgcPojo;
            }
            /** Supprime une DRGC */

            public Integer deleteDrgc(Integer drgcId) {
                String query = "Delete From T_DRGC where id=?";
                int numb = 0;
                try {
                    Connection connection = getConn();
                    PreparedStatement pstmt = connection.prepareStatement(query);
                    // Id DRGC vers requete SQL
                    pstmt.setInt(1, drgcId);// affecte la valeur du paramètre
                    // Execute la requête et retourne le nombre d'enregistrements
        supprimés
                    numb = pstmt.executeUpdate();
                    pstmt.close();
                    connection.close();
                }
                catch (Exception e) {
                    System.err.println("SQL Statement Error: " + query);
                    e.printStackTrace();
                }

                // retourne la liste
                return numb;
            }

            /** Affecte le resultSet dans le Pojo */
```

```
    private DrgcPojo[] populate(ResultSet rst) {
        DrgcPojo[] tableau = null;
        try {
                // Crée et initialise l'objet Cached RowSet
                OracleCachedRowSet result = new OracleCachedRowSet();
                result.populate(rst);
                // initialise l'objet à la taille requise
                int rsSize = getResultSetSize(result);
                // dimensionne le tableau
                tableau = new DrgcPojo[rsSize];
                int i = 0;
                while (result.next()) {
                        // initialisation de chaque objet
                        tableau[i] = new DrgcPojo();

                        tableau[i].setId(result.getInt(1));
                        tableau[i].setNom(result.getString(2));
                        i++;
                }
                // clean up
                result.close();

        } catch (Exception e) {

                System.err.println("Error populate");
                e.printStackTrace();
        }
        return tableau;
    }

/** Insère une drgc */
    public void insertDrgc (String drgcName) {
        String query = "Insert into T_DRGC (id,nom) VALUES
(SEQUENCE_DRGC_ID.nextval,?)";
        try {
                Connection connection = getConn();
                PreparedStatement pstmt = connection.prepareStatement(query);
                pstmt.setString(1, drgcName);
                pstmt.execute();
```

```
                pstmt.close();
                connection.close();
            }
            catch (Exception e) {
                System.err.println("SQL Statement Error: " + query);
                e.printStackTrace();
            }
        }


    /** Recherche une DRGC */

        public String rechercheDrgc(Integer drgcId) throws Exception {
            String query = "Select NOM From T_DRGC where id=?";
            // préparation pour le transport rpc
            DrgcPojo[] drgcPojo = null;
            try {
                Connection connection = getConn();
                PreparedStatement pstmt = connection.prepareStatement(query);
                ResultSet rst = pstmt.executeQuery();
                // Affecte le resultset dans le Pojo
                drgcPojo = populate(rst);
                rst.close();
                connection.close();
            }
            catch (Exception e) {
                System.err.println("SQL Statement Error: " + query);
                e.printStackTrace();
                throw new Exception(e.getMessage());
            }

            // retourne le résultat
            return drgcPojo;
        }
    }
```

On remarquera que la classe DAO_Drgc étend la classe abstraite DB_Conn[44] qui décrit les actions de connexions à la base de données.

[44] Voir le code de la classe DB_Conn en annexe 18

Lorsqu'une requête est exécutée, elle est au départ initialisée, parfois avec un passage de paramètre venant de la classe « appelante », puis exécutée et retourne éventuellement un résultat.
Dans l'exemple ci-dessus, les méthodes *getAllDrgc* et *rechercheDrgc* retourne un résultat, alors que les méthodes *deleteDrgc* et *insertDrgc* ne renvoient rien (elles sont identifiées par le terme « void »).

Les erreurs d'accès à la base sont gérées par les exceptions :

```
try {
        Connection connection = getConn();
        PreparedStatement pstmt = connection.prepareStatement(query);
        ResultSet rst = pstmt.executeQuery();
        // Affecte le resultset dans le Pojo
        drgcPojo = populate(rst);
        rst.close();
        connection.close();
}

catch (Exception e) {
        System.err.println("SQL Statement Error: " + query);
        e.printStackTrace();
        throw new Exception(e.getMessage());
}
```

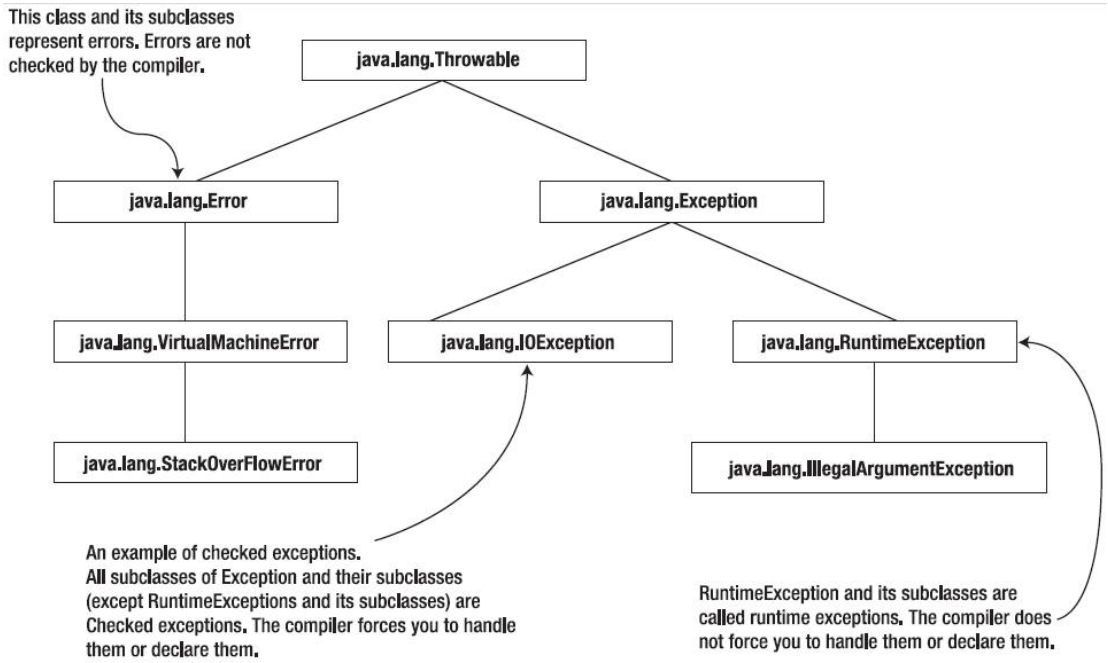
Une exception est une erreur qui se produit à l'exécution du programme sur une action de l'utilisateur ou de l'environnement. Le langage JAVA permet de traiter ces exceptions par l'utilisation d'un bloc « try » et d'un bloc « catch ».
Dans le bloc « try » on définit toutes les erreurs qui pourraient se produire et on fait suivre par un bloc « catch » dans lequel seront traitées les exceptions. Si on ne souhaite pas traiter l'exception dans ce bloc « catch », il est possible de la renvoyer à un niveau supérieur en utilisant le terme « trhow ».

Le programmeur définit lui même l'action à entreprendre pour traiter l'erreur (afficher un message d'erreur personnalisé ou attendre et recommencer) mais si aucune action n'est définie, un message d'erreur standard sera renvoyé.

La gestion des exceptions est visible dans la classe DB_Conn (la classe DAO_Drgc étend la classe DB_Conn) par les importations des classes :

import java.io.IOException;

import java.sql.SQLException;



Le fichier ***MorseServiceImpl.java*** décrit les différentes méthodes d'appel des actions sur tous les objets de données de l'application.

L'exemple ci-dessous ne reprend pas la totalité des méthodes de l'application.

Il représente le « manager » de la partie « modèle ».

```
package fr.gouv.douane.morse.server;

import com.google.gwt.user.server.rpc.RemoteServiceServlet;

import fr.gouv.douane.morse.client.MorseService;
import fr.gouv.douane.morse.shared.AgentPojo;
import fr.gouv.douane.morse.shared.DrgcPojo;
import fr.gouv.douane.morse.shared.EquipagePojo;
import fr.gouv.douane.morse.shared.NavirePojo;
import fr.gouv.douane.morse.shared.QualifPojo;

public class MorseServiceImpl extends RemoteServiceServlet implements
```

```
        MorseService {
private static final long serialVersionUID = 1L;




/* Recherche du matricule */
public AgentPojo RechercheMatricule(String matricule) {
        DAO_Agent db = new DAO_Agent();
        AgentPojo[] agentPojo = db.getSearchMatricule(matricule);

        if (agentPojo == null) {
                return null;
        } else {
                return agentPojo[0];
        }
}

/* Chargement de toutes les DRGC */
public DrgcPojo[] getAllDrgc() {
        DAO_Drgc db = new DAO_Drgc();
        DrgcPojo[] drgcPojo = db.getAllDrgc();

        if (drgcPojo == null) {
                return null;
        } else {
                return drgcPojo;
        }

}

/* Suppression d'une DRGC */

public Integer deleteDrgc(Integer drgcId) {
        DAO_Drgc db = new DAO_Drgc();
        return db.deleteDrgc(drgcId);
}

/* Insertion d'une DRGC */
```

```
	public Integer insertDrgc(String drgcName) {
		DAO_Drgc db = new DAO_Drgc();
		db.insertDrgc(drgcName);
		return 1;
	}



	/* Chargement de tous les équipages */

	public EquipagePojo[] getAllEquipage() {
		DAO_Equipage db = new DAO_Equipage();
		EquipagePojo[] equipagePojo = db.getAllEquipage();

		if (equipagePojo == null) {
			return null;
		} else {
			return equipagePojo;
		}
	}

	/* Suppression d'un type d'équipage */

	public Integer deleteEquipage(Integer equipageId) {
		DAO_Equipage db = new DAO_Equipage();
		return db.deleteEquipage(equipageId);
	}


}
```

Pour exemple la méthode *getAllDrgc()* retourne un objet de type POJO et est appelée par la classe *pageDrgc*.

```
/* Chargement de toutes les DRGC */
public DrgcPojo[] getAllDrgc() {
        DAO_Drgc db = new DAO_Drgc();
        DrgcPojo[] drgcPojo = db.getAllDrgc();

        if (drgcPojo == null) {
                return null;
        } else {
                return drgcPojo;
        }

}
```

Le code ci-dessus représente le chargement de toutes les DRGC de la table T_DRGC.
La méthode est définie comme *public*, elle est donc accessible par toutes les classes, et est de type *DrgcPojo[ ]*.
On instancie une connexion à la base de données de type *DAO_Drgc* et on alimente l'objet POJO grâce à la commande *db.getAllDrgc()* définie dans la classe *DAO_Drgc()*.
On vérifie que le POJO n'est pas vide et on retourne l'objet *drgcPojo*.
A l'affichage sur la page de gestion des DRGC, on verra apparaître toutes les DRGC présentes dans la base de données.

La partie « contrôleur ».

Le fichier **MorseService.java** est une interface qui étend le RemoteService (service d'accès à distance). Il définit toutes les méthodes utilisées dans l'application et les paramètres de chacune d'entre elles.

```
package fr.gouv.douane.morse.client;

import com.google.gwt.user.client.rpc.RemoteService;
import fr.gouv.douane.morse.shared.AgentPojo;
import fr.gouv.douane.morse.shared.DrgcPojo;
import fr.gouv.douane.morse.shared.EquipagePojo;
import fr.gouv.douane.morse.shared.NavirePojo;
import fr.gouv.douane.morse.shared.QualifPojo;

public interface MorseService extends RemoteService {
    public AgentPojo RechercheMatricule(String matricule);

    public DrgcPojo[] getAllDrgc();
    public Integer deleteDrgc(Integer drgcId);
    public Integer insertDrgc(String drgcName);

    public EquipagePojo[] getAllEquipage();
    public Integer deleteEquipage(Integer equipageId);
    public Integer insertEquipage(Integer nbAgent, String drgcName, Integer code);

    public NavirePojo[] getAllNavire() throws Exception ;
    public Integer deleteNavire(Integer navireId) throws Exception ;
    public Integer insertNavire(String nom, Integer longueur, Integer capacite, Integer
drgcId, String numCodique, Integer typeEquipageId) throws Exception ;
    public String rechercheDrgc(Integer drgcId) throws Exception;

    public QualifPojo[] getAllQualif() throws Exception ;
    public Integer deleteQualif(Integer qualifId) throws Exception ;
    public Integer insertQualif(String libelle, Integer periode, Integer
reglementationId, String type, Integer orgaDelivrance, Integer orgaValidation) throws
Exception ;
}
```

Ce fichier fonctionne en parallèle avec le fichier **MorseServiceAsync.java** qui définit les actions de « callback » de l'application.

Lorsqu'une requête est envoyée au serveur, le « callback » permet de traiter la requête tout en continuant de traiter les autres tâches. Quand le résultat est prêt, un « callback » sera émis pour « reprendre la main ».
L'exemple ci-dessous ne reprend pas toutes les descriptions de « callback » de l'application MORSE.

```
package fr.gouv.douane.morse.client;

import com.google.gwt.user.client.rpc.AsyncCallback;
import fr.gouv.douane.morse.shared.AgentPojo;
import fr.gouv.douane.morse.shared.DrgcPojo;
import fr.gouv.douane.morse.shared.EquipagePojo;
import fr.gouv.douane.morse.shared.NavirePojo;
import fr.gouv.douane.morse.shared.QualifPojo;

public interface MorseServiceAsync {

    // Agent
    public void RechercheMatricule(String matricule, AsyncCallback<AgentPojo>
callback);

    // DRGC
    public void getAllDrgc(AsyncCallback<DrgcPojo[]> callback);
    public void deleteDrgc(Integer drgcId, AsyncCallback<Integer> callback);
    public void insertDrgc(String drgcName, AsyncCallback<Integer> callback);


    // Equipage
    public void getAllEquipage(AsyncCallback<EquipagePojo[]> callback);
    public void deleteEquipage(Integer equipageId, AsyncCallback<Integer>
callback);
    public void insertEquipage(Integer nbAgent, String drgcName, Integer code,
AsyncCallback<Integer> callback);


}
```

Ces deux fichiers sont repris dans le répertoire *client* de l'application MORSE, ils sont le pendant du fichier *MorseServiceImpl.java* côté « client ».
La connexion RPC[45] est décrite dans le chapitre 3.4.3 ci-dessous.

La partie « vue ».

Elle permet d'afficher à l'écran les données qui sont requises par l'utilisateur. L'exemple donné ci-dessous permet l'affichage de la fonctionnalité *GESTION DRGC*.
Le fichier PageDrgc, repris dans le répertoire « client » est l'une des fonctionnalités de l'application MORSE.
Cette fonctionnalité permet à l'utilisateur d'ajouter ou de supprimer une DRGC.

```
package fr.gouv.douane.morse.client.drgc;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.rpc.ServiceDefTarget;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.FlexTable;
import com.google.gwt.user.client.ui.HasVerticalAlignment;
import com.google.gwt.user.client.ui.HorizontalPanel;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.VerticalPanel;
import fr.gouv.douane.morse.client.MorseService;
import fr.gouv.douane.morse.client.MorseServiceAsync;
import fr.gouv.douane.morse.client.PageAccueil;
import fr.gouv.douane.morse.shared.DrgcPojo;

/**Entry point classes define <code>onModuleLoad()</code> */

public class PageDrgc extends Composite {
```

[45] RPC : Remote Procedure Call

```
// ***************** Debut initialisation ********************
private FlexTable drgcFlexTable = new FlexTable();
private TextBox drgcTextBox = new TextBox();
private Button addDrgcButton = new Button("Ajouter");
private HorizontalPanel drgcPanel = new HorizontalPanel();
final VerticalPanel drgcCodeWidg = new VerticalPanel();
final VerticalPanel drgcTextWidg = new VerticalPanel();
final VerticalPanel drgcWidg = new VerticalPanel();

/** Définition pour le panel d'ajout */
final Label drgcTextLabel = new Label("Nom");
VerticalPanel hPanel = new VerticalPanel();
Label titre = new Label("");
Button sendButtonR = new Button("RETOUR");
private PageAccueil pa;
int k = 0;
// ***************** Fin initialisation ********************

/** Page de présentation des DRGC */

public PageDrgc(final DrgcPojo[] drgcPojo) {

        titre.setText("GESTION DRGC");
        titre.setStyleName("titre");
        drgcWidg.add(titre);

        // création de la table flexible
        // Formattage de la table flexible.
        drgcFlexTable.setBorderWidth(2);
        drgcFlexTable.setCellPadding(6);
        drgcFlexTable.getRowFormatter().addStyleName(0, "qualifListHeader");
        drgcFlexTable.addStyleName("qualifList");
        drgcFlexTable.getColumnFormatter().addStyleName(0, "qualifLibelle");
        drgcFlexTable.setText(0, 0, "Nom");
        drgcFlexTable.setText(0, 1, "Suppression");

        // Boucle pour alimenter le tableau + Ajout d'un bouton de suppression drgc
        final int indDrgcFlexTableR = drgcPojo.length;
        for (k = 0; k < indDrgcFlexTableR; k++) {
                // alimentation de la drgc
```

```
                    drgcFlexTable.setText(k + 1, 0, drgcPojo[k].getNom());
                    drgcFlexTable.getCellFormatter().addStyleName(k + 1,
0,"qualifListRemoveColumn");

                    // ajouter un bouton de suppression
                    final Button removeDrgcButton = new Button("x");
                    removeDrgcButton.setTitle(drgcPojo[k].getId().toString());
                    removeDrgcButton.addStyleDependentName("remove");
                    drgcFlexTable.getCellFormatter().addStyleName(k + 1,
1,"qualifListRemoveColumn");
                    drgcFlexTable.setWidget(k + 1, 1, removeDrgcButton);
                    removeDrgcButton.addClickHandler(new ClickHandler() {
                            public void onClick(ClickEvent event) {

                                    Integer ind =
Integer.parseInt(removeDrgcButton.getTitle());

                                    callServerDelete(ind);
                            }
                    });

            } // Fin ajout bouton suppression drgc

            // Assemblage du panel d'ajout.
            drgcTextWidg.add(drgcTextLabel);
            drgcTextWidg.add(drgcTextBox);
            drgcPanel.add(drgcCodeWidg);
            drgcPanel.add(drgcTextWidg);
            drgcPanel.add(addDrgcButton);

drgcPanel.setCellVerticalAlignment(addDrgcButton,HasVerticalAlignment.ALIGN_B
OTTOM);



            // Assemblage du panel principal.
            drgcWidg.setSpacing(15);
            drgcWidg.add(drgcFlexTable);
            drgcWidg.add(drgcPanel);
            drgcWidg.add(sendButtonR);
```

```
		// Focus du curseur sur la boite d'entrée.
		drgcTextBox.setFocus(true);

		// Ajout d'une qualif lors du click sur le bouton ajouter.
		addDrgcButton.addClickHandler(new ClickHandler() {
			public void onClick(ClickEvent event) {
				addDrgcButton.setEnabled(false);
				final String symbolDrgc =
drgcTextBox.getText().toUpperCase().trim();
				callServerInsert(symbolDrgc);
				drgcTextBox.setText("");
				addDrgcButton.setEnabled(true);
				initWidget(drgcWidg);
			}
		});

		// ***************** Debut Bouton ********************
		// Retour à la page d'accueil.
		sendButtonR.addClickHandler(new ClickHandler() {
			public void onClick(ClickEvent event) {
				RootPanel.get().remove(0);
				pa = new PageAccueil();
				RootPanel.get().add(pa);
				Window.alert("Accueil");
			}
		});

		// ***************** Fin Bouton ********************
		// Affiche la page DRGC
		initWidget(drgcWidg);
	}



	/** Appel au serveur pour supprimer une DRGC   */
	public void callServerDelete(Integer drgcId) {
		// définit le service à appeler
		MorseServiceAsync svc = (MorseServiceAsync)
GWT.create(MorseService.class);
```

```
                ServiceDefTarget endpoint = (ServiceDefTarget) svc;
                endpoint.setServiceEntryPoint(GWT.getModuleBaseURL() + "morse");
                svc.deleteDrgc(drgcId, new AsyncCallback<Integer>() {
                        public void onFailure(Throwable caught) {
                                // affichage du message d'erreur
                                Window.alert("cette DRGC ne peut pas etre supprimee,
des agents y sont affectes");
                        }

                        public void onSuccess(Integer result) {
                                pa = new PageAccueil();
                                pa.callServer();
                        }
                });
        }

        /**  Insérer une DRGC */

        public void callServerInsert(String drgcName) {
                // définit le service à appeler
                MorseServiceAsync svc = (MorseServiceAsync) GWT
                                .create(MorseService.class);
                ServiceDefTarget endpoint = (ServiceDefTarget) svc;
                endpoint.setServiceEntryPoint(GWT.getModuleBaseURL() + "morse");
                svc.insertDrgc(drgcName, new AsyncCallback<Integer>() {
                        public void onFailure(Throwable caught) {
                                // message d'erreur
                                Window.alert("cette DRGC existe deja");
                        }

                        public void onSuccess(Integer result) {
                                pa = new PageAccueil();
                                pa.callServer();
                        }
                });
        }
}
```

La classe *PageDrgc* étend la classe *Composite* qui permet de définir des objets qui contiennent des objets. Ainsi, les actions (dessiner, redimensionner, etc ...) peuvent

être effectuées sans que l'on ait à définir si l'objet est « composite » ou « primitif » (ex : un ensemble de lignes, de carrés, de ronds).

Cette classe met en œuvre l'utilisation d'un tableau flexible, selon le nombre d'enregistrements qui se trouve dans la table T_DRGC.

Au dessous de cette table flexible, un panneau d'ajout permet de rajouter un élément dans la table.

Les actions d'ajout et de suppression s'effectuent au clic sur le bouton « valider » et « supprimer ».

3.3.2.3. Les exemples de programmation en JAVA de l'application MORSE

L'application MORSE a été développée dans le respect du CDCF et du palier technique de la DGDDI. Les exemples ci-dessous montrent le résultat obtenu à partir du code JAVA de l'application.

La page d'accueil :

```
package fr.gouv.douane.morse.client;

import com.google.gwt.core.client.GWT;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.google.gwt.user.client.rpc.ServiceDefTarget;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.Composite;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

import fr.gouv.douane.morse.client.agent.PageMatricule;
import fr.gouv.douane.morse.client.drgc.PageDrgc;
import fr.gouv.douane.morse.client.equipage.PageEquipageC;
import fr.gouv.douane.morse.client.etat.PageEtat;
import fr.gouv.douane.morse.client.navire.PageNavire;
import fr.gouv.douane.morse.client.qualification.PageQualification;
import fr.gouv.douane.morse.client.reglementation.PageReglementation;
```

```
import fr.gouv.douane.morse.shared.DrgcPojo;
import fr.gouv.douane.morse.shared.EquipagePojo;
import fr.gouv.douane.morse.shared.NavirePojo;
import fr.gouv.douane.morse.shared.QualifPojo;

/** Entry point classes define <code>onModuleLoad()</code> */

public class PageAccueil extends Composite {

    // Définition des boutons de la page d'accueil
    private Button sendButton = new Button("Gestion Agent");
    private Button sendButton1 = new Button("Gestion Equipage");
    private Button sendButton2 = new Button("Gestion Navire");
    private Button sendButton3 = new Button("Gestion DRGC");
    private Button sendButton4 = new Button("Gestion Reglementation");
    private Button sendButton5 = new Button("Gestion Qualification");
    private Button sendButton6 = new Button("Edition des Etats");
    private Button sendButton7 = new Button("Parametrage");

    // Définition des pages menus
    private PageMatricule pm;
    private PageEquipageC pec;
    private PageNavire pn;
    private PageDrgc pd;
    private PageReglementation pr;
    private PageQualification pq;
    private PageEtat pe;
    VerticalPanel hPanel = new VerticalPanel();

    // Définition des Pojos
    NavirePojo[] navirePojo;
    DrgcPojo[] drgcPojo;
    EquipagePojo[] equipagePojo;

    public PageAccueil() {
        // Ajout de style aux boutons
        sendButton.addStyleName("sendButton");
        sendButton1.addStyleName("sendButton");
        sendButton2.addStyleName("sendButton");
        sendButton3.addStyleName("sendButton");
```

```
sendButton4.addStyleName("sendButton");
sendButton5.addStyleName("sendButton");
sendButton6.addStyleName("sendButton");
sendButton7.addStyleName("sendButton");

// Espacement entre les widgets
hPanel.setSpacing(5);
hPanel.addStyleName("verticalPanel");

// Ajout des boutons à la page d'accueil
hPanel.add(sendButton);
hPanel.add(sendButton1);
hPanel.add(sendButton2);
hPanel.add(sendButton3);
hPanel.add(sendButton4);
hPanel.add(sendButton5);
hPanel.add(sendButton6);
hPanel.add(sendButton7);

// ***************** Debut Bouton ********************
// click sur bouton Agent
sendButton.addClickHandler(new ClickHandler() {
        public void onClick(ClickEvent event) {
                RootPanel.get().remove(0);
                pm = new PageMatricule();
                RootPanel.get().add(pm);
        }
}); // FIN BOUTON AGENT

// Click sur le bouton Equipage
sendButton1.addClickHandler(new ClickHandler() {
        public void onClick(ClickEvent event) {
                sendButton1.setEnabled(false);
                callServerEquipage();
        }
}); // FIN BOUTON EQUIPAGE

// Click sur le bouton Navire
sendButton2.addClickHandler(new ClickHandler() {
        public void onClick(ClickEvent event) {
```

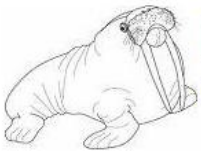
```
                        sendButton2.setEnabled(false);
                        callServerNavire();
                }
        }); // FIN BOUTON NAVIRE
        // Click sur le bouton DRGC
        sendButton3.addClickHandler(new ClickHandler() {
                public void onClick(ClickEvent event) {
                        sendButton3.setEnabled(false);
                        callServer();
                }
        }); // FIN BOUTON DRGC

        // Click sur le bouton Réglementation
        sendButton4.addClickHandler(new ClickHandler() {
                public void onClick(ClickEvent event) {
                        RootPanel.get().remove(0);
                        pr = new PageReglementation();
                        RootPanel.get().add(pr);
                }
        }); // FIN BOUTON REGLEMENTATION

        // Click sur le bouton Qualification
        sendButton5.addClickHandler(new ClickHandler() {
                public void onClick(ClickEvent event) {
                        sendButton5.setEnabled(false);
                        callServerQualif();
                }
        }); // FIN BOUTON QUALIFICATION

        // Click sur le bouton Edition
        sendButton6.addClickHandler(new ClickHandler() {
                public void onClick(ClickEvent event) {
                        RootPanel.get().remove(0);
                        pe = new PageEtat();
                        RootPanel.get().add(pe);
                }
        }); // FIN BOUTON EDITION

        // Click sur le bouton Paramètrage
        sendButton7.addClickHandler(new ClickHandler() {
```

```
                        public void onClick(ClickEvent event) {
                                Window.alert("TODO: Parametre");
                        }
                });

                // ***************** Fin Bouton ********************
                // Affiche la page d'accueil

                initWidget(hPanel);
        }

        /** affichage de la page DRGC */

        public void callServer() {
                MorseServiceAsync svc = (MorseServiceAsync) GWT
                                .create(MorseService.class);
                ServiceDefTarget endpoint = (ServiceDefTarget) svc;
                endpoint.setServiceEntryPoint(GWT.getModuleBaseURL() + "morse");

                svc.getAllDrgc(new AsyncCallback<DrgcPojo[]>() {
                        public void onFailure(Throwable caught) {
                                sendButton3.setEnabled(true);
                        }

                        public void onSuccess(DrgcPojo[] result) {
                                        RootPanel.get().remove(0);
                                        pd = new PageDrgc(result);
                                        RootPanel.get().add(pd);
                                }
                });
        }

        /** Affichage de la page Equipage *
         */
        public void callServerEquipage() {
                // define the service you want to call
                MorseServiceAsync svc = (MorseServiceAsync)
GWT.create(MorseService.class);
                ServiceDefTarget endpoint = (ServiceDefTarget) svc;
                endpoint.setServiceEntryPoint(GWT.getModuleBaseURL() + "morse");
```

```
                svc.getAllEquipage(new AsyncCallback<EquipagePojo[]>() {
                        public void onFailure(Throwable caught) {
                                // Show the RPC error message to the user
                                Window.alert("erreur Equipage");
                                sendButton1.setEnabled(true);
                        }


                        public void onSuccess(EquipagePojo[] result) {
                                        RootPanel.get().remove(0);
                                        pec = new PageEquipageC(result);
                                        RootPanel.get().add(pec);
                                }
                });
        }

        /** Affichage de la page Navire */
        public void callServerNavire() {
                final MorseServiceAsync svc = (MorseServiceAsync) GWT
                                .create(MorseService.class);
                ServiceDefTarget endpoint = (ServiceDefTarget) svc;
                endpoint.setServiceEntryPoint(GWT.getModuleBaseURL() + "morse");
                svc.getAllNavire(new AsyncCallback<NavirePojo[]>() {
                        public void onFailure(Throwable caught) {
                                // Show the RPC error message to the user
                                Window.alert("erreur Navire"
                                                +
fr.gouv.douane.morse.shared.OraErr.getErrMessage(caught.getMessage()));
                                sendButton2.setEnabled(true);
                        }

                        public void onSuccess(NavirePojo[] result) {
                                navirePojo = result;
                                svc.getAllDrgc(new AsyncCallback<DrgcPojo[]>() {
                                        public void onFailure(Throwable caught) {
                                                Window.alert("erreur Drgc"
                                                                +
fr.gouv.douane.morse.shared.OraErr.getErrMessage(caught.getMessage()));
                                                sendButton2.setEnabled(true);
                                        }
```

```
                                        public void onSuccess(DrgcPojo[] result) {
                                                drgcPojo = result;
                                                svc.getAllEquipage(new
AsyncCallback<EquipagePojo[]>() {
                                                public void onFailure(Throwable caught)
{




                                                                Window.alert("erreur
Equipage"
                                                                                +
fr.gouv.douane.morse.shared.OraErr.getErrMessage(caught.getMessage()));

      sendButton2.setEnabled(true);
                                                        }

                                                        public void
onSuccess(EquipagePojo[] result) {
                        RootPanel.get().remove(0);
                        equipagePojo = result;
                        pn = new PageNavire(navirePojo, drgcPojo, equipagePojo);
                        RootPanel.get().add(pn);
                }

      });
      }
});
}
});
}

      /** Affichage de la page Qualification */
      public void callServerQualif() {
            MorseServiceAsync svc = (MorseServiceAsync)
GWT.create(MorseService.class);
            ServiceDefTarget endpoint = (ServiceDefTarget) svc;
            endpoint.setServiceEntryPoint(GWT.getModuleBaseURL() + "morse");
            svc.getAllQualif(new AsyncCallback<QualifPojo[]>() {
```

```
                        public void onFailure(Throwable caught) {
                                Window.alert("erreur Qualification");
                                sendButton5.setEnabled(true);
                        }
                        public void onSuccess(QualifPojo[] result) {
                                        RootPanel.get().remove(0);
                                        pq = new PageQualification(result);
                                        RootPanel.get().add(pq);
                                }
                });
        }
}
```



MORSE

Module d'Organisation des Ressources Spécialisées et des Equipages

- Gestion Agent
- Gestion Equipage
- Gestion Navire
- Gestion DRGC
- Gestion Reglementation
- Gestion Qualification
- Edition des Etats
- Parametrage

Au clic sur le bouton « Gestion DRGC », la page s'affiche comme suit, selon le code programme donné en exemple dans le chapitre 3.3.2.2 ci-dessus.

GESTION DRGC

Nom	Suppression
Antilles Guyane	x
Atlantique	x
Manche Mer du Nord	x
MEDITERRANNEE	x

Nom

Ajouter

RETOUR

3.3.3. Oracle 10g une base de données pour MORSE

3.3.3.1. Présentation de la base de données

De même que pour le langage JAVA, la base de données implémentée sur Oracle 10g appartient au palier technique de la DGDDI.

Une base de données Oracle 10g est un ensemble de données traité comme une unité. L'objectif d'une base de données est de stocker et de récupérer des informations qui ont des relations entre elles. Un serveur de base de données permet de gérer un grand nombre de données dans un environnement multi-utilisateurs avec des accès concurrentiels tout en fournissant une bonne performance. Un serveur de donnée permet également d'assurer la sécurité d'accès aux données et de fournir des solutions de récupération des données en cas de panne.

Les bases de données Oracle sont les premières élaborées pour les « grilles de calcul » au sein d'entreprises.
La «grille de calcul» (ou «grid computing» en anglais), est un terme utilisé pour qualifier la disponibilité de grandes quantités de ressources (calcul, stockage, services et applications) sur les réseaux informatiques. Ainsi, le principe repose sur la recherche à travers le réseau mondial de « puissance de calcul » sur tous les ordinateurs inter-connectés. Le «grid computing» est au cœur d'enjeux économiques dans de nombreux secteurs : l'énergie, la finance, les biotechnologies et les grands industriels.

Une base de données a une structure logique et une structure physique indépendantes l'une de l'autre autorisant ainsi une gestion du stockage physique des données sans conséquences pour l'accès logique aux structures de stockage.
Les systèmes d'information construits sur le modèle «grid computing» offrent une plus grande qualité de service (pas de point unique de panne, une infrastructure de sécurité performante et centralisée, une gestion selon des règles), à coût réduit (utilisation des ressources et réduction drastique des coûts de gestion et de maintenance) et une plus grande flexibilité (les ressources sont mutualisées

permettant d'éliminer la sous-utilisation et la redondance des ressources disponibles tout en assurant une plus grande flexibilité des ressources selon les besoins).
Les ressources gérées à travers un «grid computing» incluent :

- l'infrastructure : les matériels et les logiciels qui créent une banque de données et les programmes d'exécution de l'environnement
- les applications : les flux et programmes qui définissent un processus métier sépcifique
- l'information : les définitions de tous les types de données utilisés dans un processus métier.

La famille de base de données Oracle 10g de type «grid computing» fournit une structure de sécurité fiable, une gestion centralisée, une intuitivité, des outils de développement puissants et un accès universel.

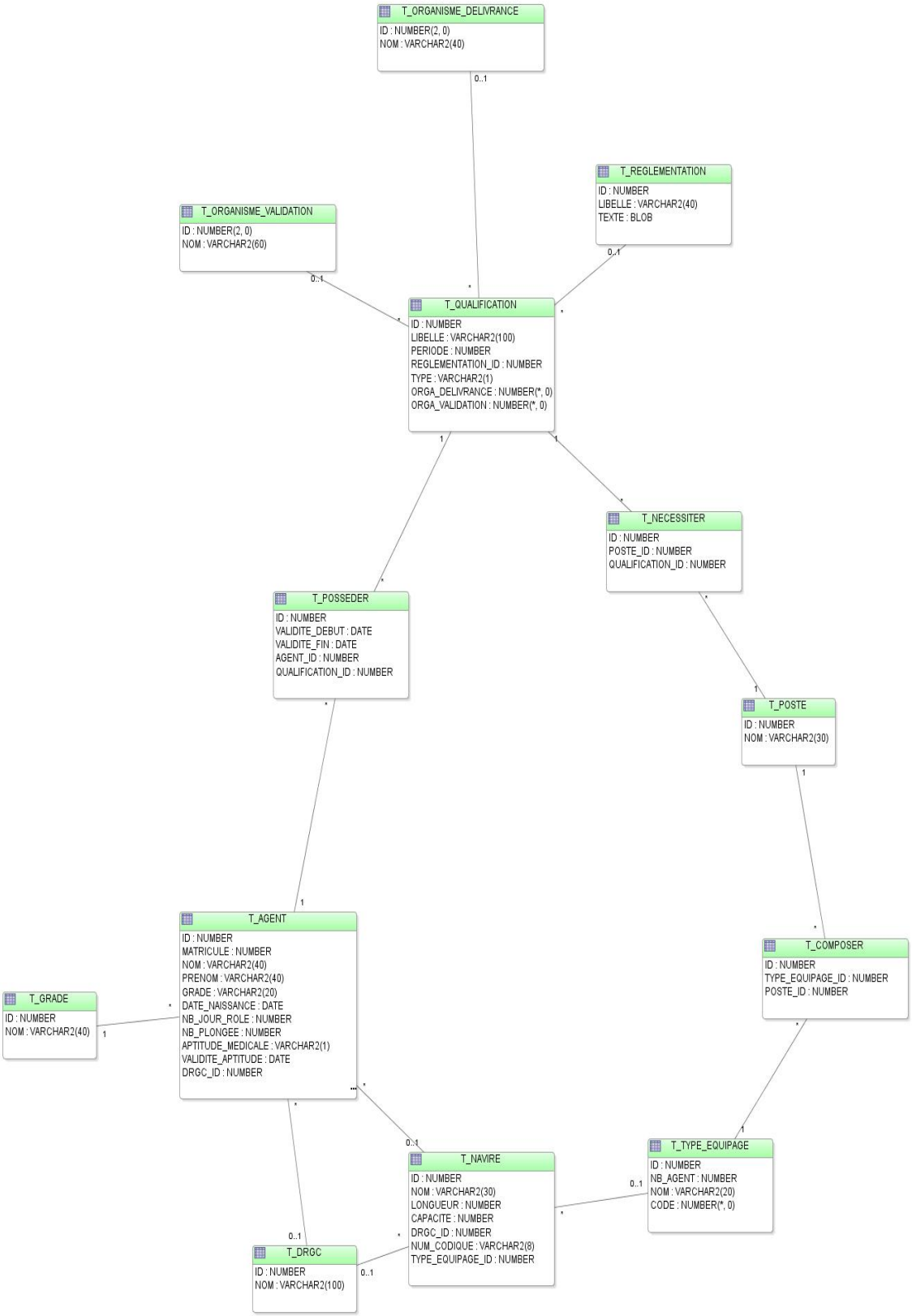
Les données reprises dans la base de données MORSE ne sont pas conséquentes en terme de volume. En application locale (uniquement pour la DRGC-MMN), elle représente 149 agents, 5 navires, 16 qualifications, 27 réglementations et 4 types d'équipage soit environs 50 000 octets.
Lorsque l'application sera généralisée aux autres DRGC, le volume de données progressera de 400%.

La base de données reprend les tables suivantes :
T_AGENT : données sur les agents
T_NAVIRE : données sur les navires
T_TYPE_EQUIPAGE : données sur le type d'équipage
T_POSTE : données sur le poste occupé sur le navire
T_REGLEMENTATION : données sur les réglementations en vigueur
T_DRGC : données sur les Directions Régionales Garde-Côtes
T_QUALIFICATION : données sur les qualifications et formations
T_COMPOSITION : données sur la composition d'un équipage
T_REQUIS : données sur la qualification requise pour un poste
T_AGENT_QUALIFICATION : données sur les qualifications que possède un agent
T_GRADE : données sur le grade d'un agent

T_ORGANISME_DELIVRANCE : données sur les organismes de délivrance des qualifications et formations

T_ORGANISME_VALIDATION : données sur les organismes de validation des qualifications et formations.

Le modèle de données ci-dessous représente l'ensemble des tables de la base de données MORSE



Pour chacune des tables, un dictionnaire de données a été défini dans le CDCT ainsi qu'un modèle de données qui permet de déterminer les relations entre elles.

Toutes les données du projet MORSE ont été définies et dimensionnées à partir des besoins de la MOA (CDCF du projet MORSE).

Elles sont de type numérique, alphanumérique, date et LOB[46].

Le type LOB est utilisé dans ce projet pour stocker, en format PDF, les textes réglementaires qui régissent le processus métier du projet MORSE.

3.3.3.2. Implémentation de la base de données

L'installation de la base de données Oracle 10g ne présente pas de difficultés particulières. Volontairement je ne décrirai pas l'installation à partir du fichier « OracleXE_new.exe» téléchargé sur le site Oracle. Le guide d'installation est également donné et le programme d'installation est un auto-exécutable qu'il suffit de suivre pas à pas.

Les pré-requis à l'installation sont :

Processeur	Intel (x86)
Système d'exploitation	•Windows 2000 (SP4 ou +) •Windows Server 2003 •Windows XP Professional (SP1 ou +)
Protocole réseau	TCP/IP
Espace disque	Serveur : 1.6 gigabytes minimum Client : 75 megabytes
Mémoire (RAM)	256 megabytes minimum, 512 megabytes recommandés
Microsoft Windows Installer (MSI)	MSI version 2.0 ou plus
Navigateur internet	Microsoft Internet Explorer 6.0 ou plus •Netscape Navigator 7.2 ou plus •Mozilla 1.7 ou plus •Firefox 1.0 ou plus

[46] LOB : Large Object ou BLOB (Binary Large Object) est un type de données permettant de stocker des informations dans une base de données, tels que les images, le son ou la video

La configuration de mon poste de travail (poste développeur) est :

Processeur	Intel Pentium Dual Core
Système d'exploitation	Windows Vista
Protocole réseau	TCP/IP
Espace disque	150 gigabytes
Mémoire (RAM)	2 gigabytes
Microsoft Windows Installer (MSI)	MSI version 2.0 ou plus
Navigateur	Firefox 3.6.8

L'installation effectuée comprend un serveur et un client de la base de données Oracle 10g. En développement, le serveur et le client sont physiquement sur le poste développeur, généralement, le client se trouve sur un poste différent de celui du serveur.

Pour effectuer des opérations sur Oracle, le système procède de la manière suivante :

- une instance[47] démarre sur l'ordinateur sur lequel se trouve le serveur Oracle (serveur hôte)
- l'ordinateur (différent du serveur) sur lequel se trouve une application démarre un processus utilisateur qui cherche à établir une connexion au serveur à l'aide du pilote adéquat
- le serveur détecte la tentative de connexion de l'application, à l'aide du pilote adéquat et crée un processus serveur dédiée à ce processus utilisateur
- l'utilisateur effectue une requête SQL[48]
- le serveur vérifie que la requête appartient à un espace SQL partagé, s'il le trouve, le serveur vérifie les habilitations utilisateur et l'espace SQL partagé est utilisé pour satisfaire la requête, sinon le serveur créera un espace SQL partagé pour traiter la requête
- le serveur récupère les données stockées dans la base de données ou dans le SGA[49] et afférentes à la requête, ou il procède aux modifications demandées (modification de données, suppression d'enregistrement, etc ....) tout en assurant

[47] Voir annexe 19 : instance Oracle

[48] SQL : Structured Query Langage

[49] SGA : System Global Area

une «sauvegarde» en cas de rupture de liaison ou d'opération incomplète entre le serveur et le client

●si l'opération est correcte, le serveur envoie un message au client pour l'en avertir, sinon c'est un message d'erreur qui est envoyé

●le gestionnaire de base de données vérifie qu'aucune intervention n'est requise (opération incorrecte, retour arrière, ...) et gère aussi l'accès concurrentiel aux données.

Après l'installation d'Oracle 10g, il s'agissait de créer les différentes tables présentées dans le chapitre précédent.

L'utilisation du navigateur Firefox avec pour adresse du serveur « 127.0.0.1 » donne accès à une page de connexion sécurisée à la base de données avec login et mot de passe.

Puis Oracle 10g offre les fonctionnalités d'administration de la base, de gestion des tables, de requêtes SQL et d'utilitaires tels que l'import, l'export, génération de DLL[50].

J'ai commencé par utiliser les fonctionnalités de base d'Oracle 10g pour créer les tables de l'application MORSE et me familiariser avec la création de ces tables sous Oracle mais les fonctionnalités étant trop restreintes et très peu intuitives, j'ai ensuite utilisé Oracle SQL developer[51] pour créer, modifier et alimenter ma base de données.

Je n'avais aucune connaissance sur SQL developer mais, ce logiciel est très proche dans ses fonctionnalités, du logiciel TOAD[52] que j'utilise régulièrement dans le cadre de mes fonctions professionnelles.

3.3.3.3. Exemple de création de tables dans la base de données

La création de tables s'est faite à l'aide du logiciel SQL developer, pour chacune des tables une clé primaire a été définie, pour certaines des clés étrangères ont été également définies.

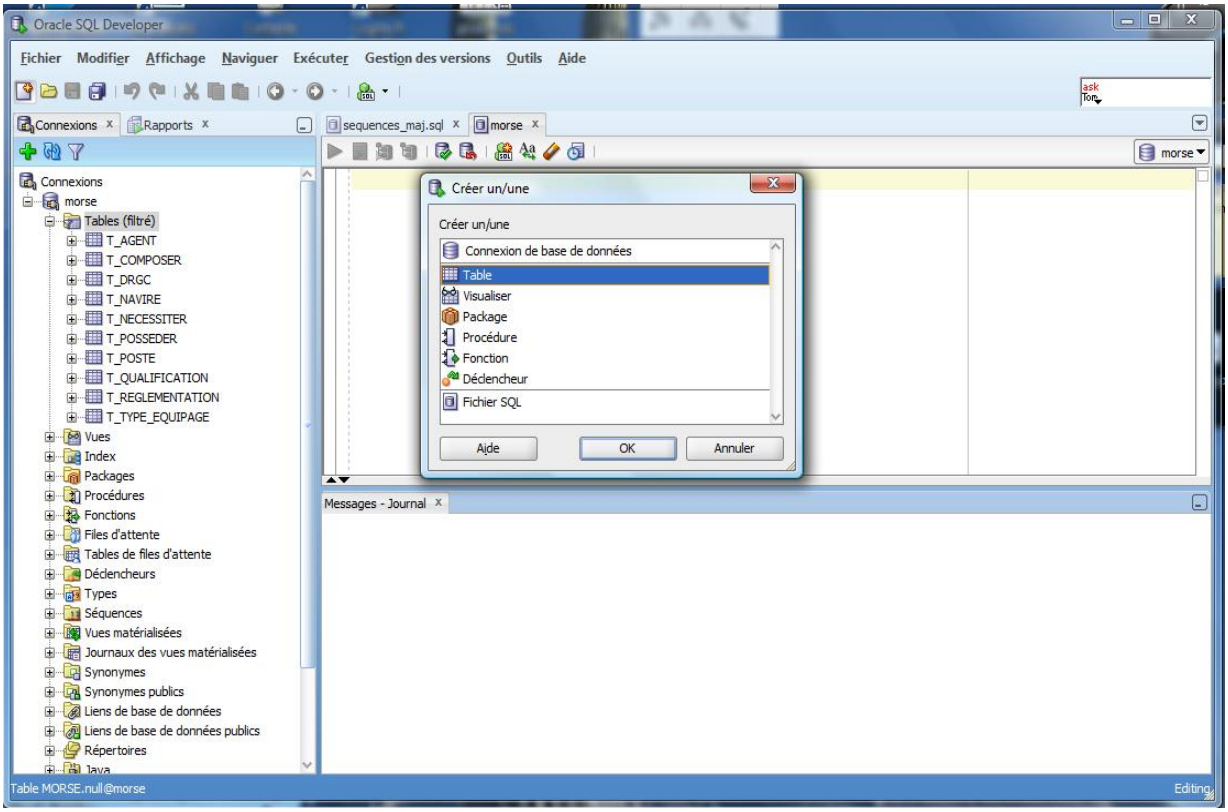
Quelques tables possèdent des contraintes exprimées sur des champs, contrainte «unique» par exemple.

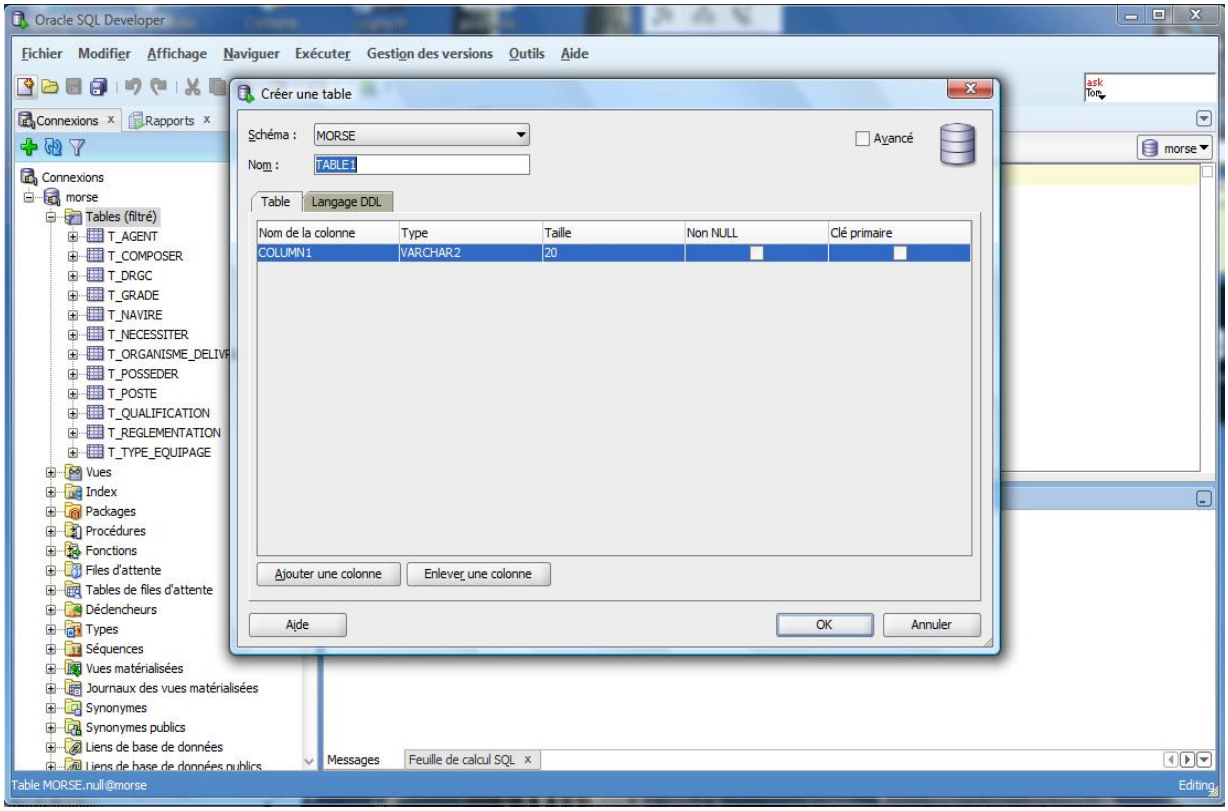
[50] DLL : Dynamic Link Library

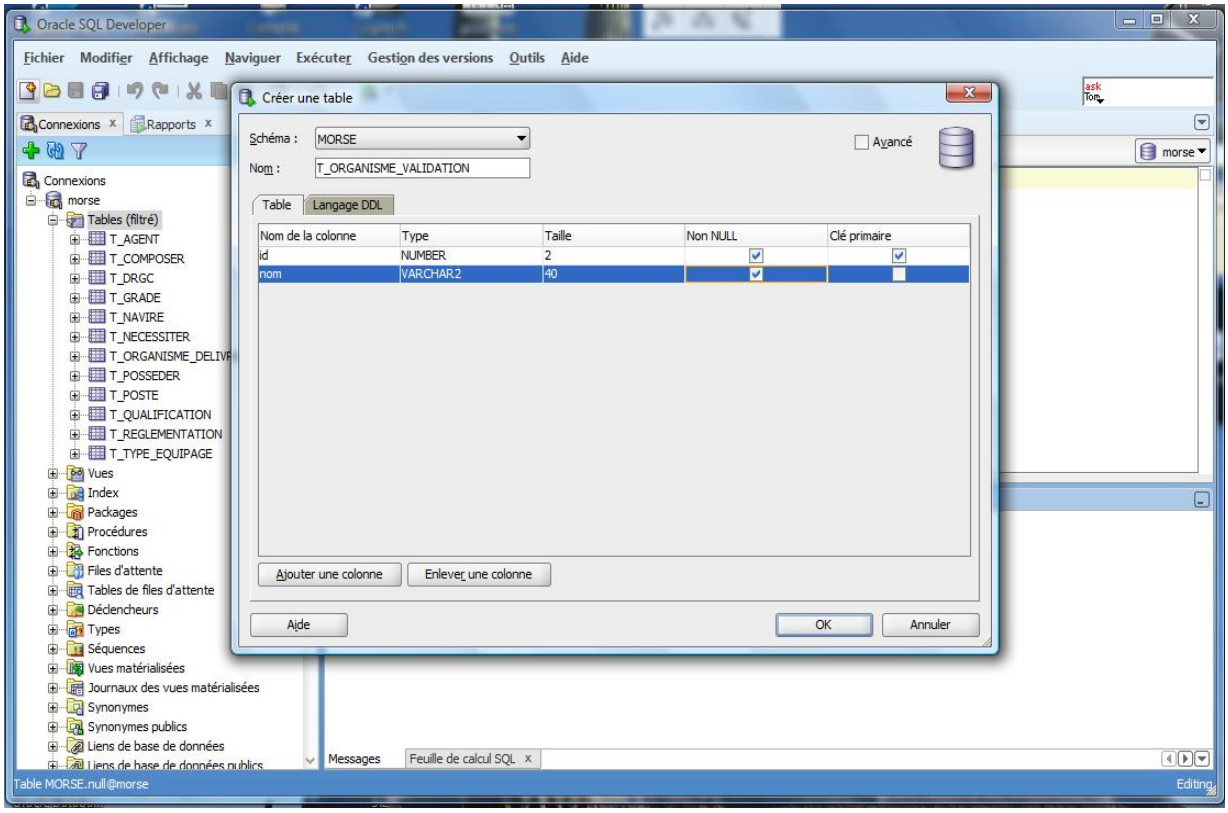
[51] Logiciel de gestion de bases de données, gratuit de ORACLE

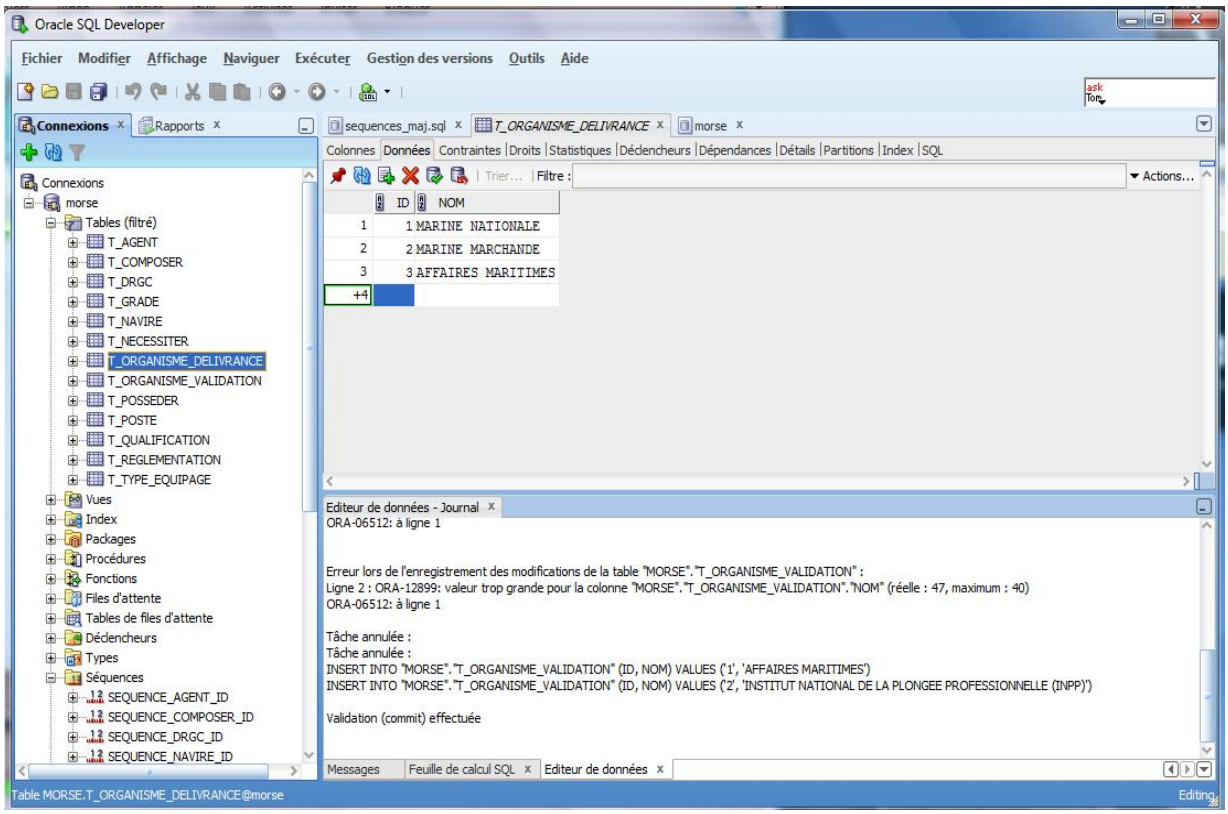
[52] Logiciel de gestion de base de données de QUEST SOFTWARE

Les écrans présentés ci-dessous montrent la création et l'alimentation de la table T_ORGANISME_VALIDATION.




Oracle SQL Developer
Fichier Modifier Affichage Naviguer Exécuter Gestion des versions Outils Aide
Créer une table
Schéma : MORSE
Nom : TABLE1
Avancé
Table Langage DDL
Nom de la colonne Type Taille Non NULL Clé primaire
COLUMN1 VARCHAR2 20
Ajouter une colonne
Enlever une colonne
Aide
OK
Annuler
Connexions
Rapports
morse
Tables (filtré)
T_AGENT
T_COMPOSER
T_DRGC
T_GRADE
T_NAVIRE
T_NECESSITER
T_POSSEDER
T_POSTE
T_QUALIFICATION
T_REGLEMENTATION
T_TYPE_EQUIPAGE
Vues
Index
Packages
Procédures
Fonctions
Files d'attente
Tables de files d'attente
Déclencheurs
Types
Séquences
Vues matérialisées
Journaux des vues matérialisées
Synonymes
Synonymes publics
Liens de base de données
Messages
Feuille de calcul SQL
Table MORSE.null@morse
Editing


Oracle SQL Developer
Fichier Modifier Affichage Naviguer Exécuter Gestion des versions Outils Aide
Créer une table
Schéma : MORSE
Nom : T_ORGANISME_VALIDATION
Avancé
Table Langage DDL
Nom de la colonne Type Taille Non NULL Clé primaire
id NUMBER 2
nom VARCHAR2 40
Ajouter une colonne
Enlever une colonne
Aide
OK
Annuler
Connexions
Rapports
morse
Tables (filtré)
T_AGENT
T_COMPOSER
T_DRGC
T_GRADE
T_NAVIRE
T_NECESSITER
T_POSSEDER
T_POSTE
T_QUALIFICATION
T_REGLEMENTATION
T_TYPE_EQUIPAGE
Vues
Index
Packages
Procédures
Fonctions
Files d'attente
Tables de files d'attente
Déclencheurs
Types
Séquences
Vues matérialisées
Journaux des vues matérialisées
Synonymes
Synonymes publics
Liens de base de données
Messages
Feuille de calcul SQL
Table MORSE.null@morse
Editing

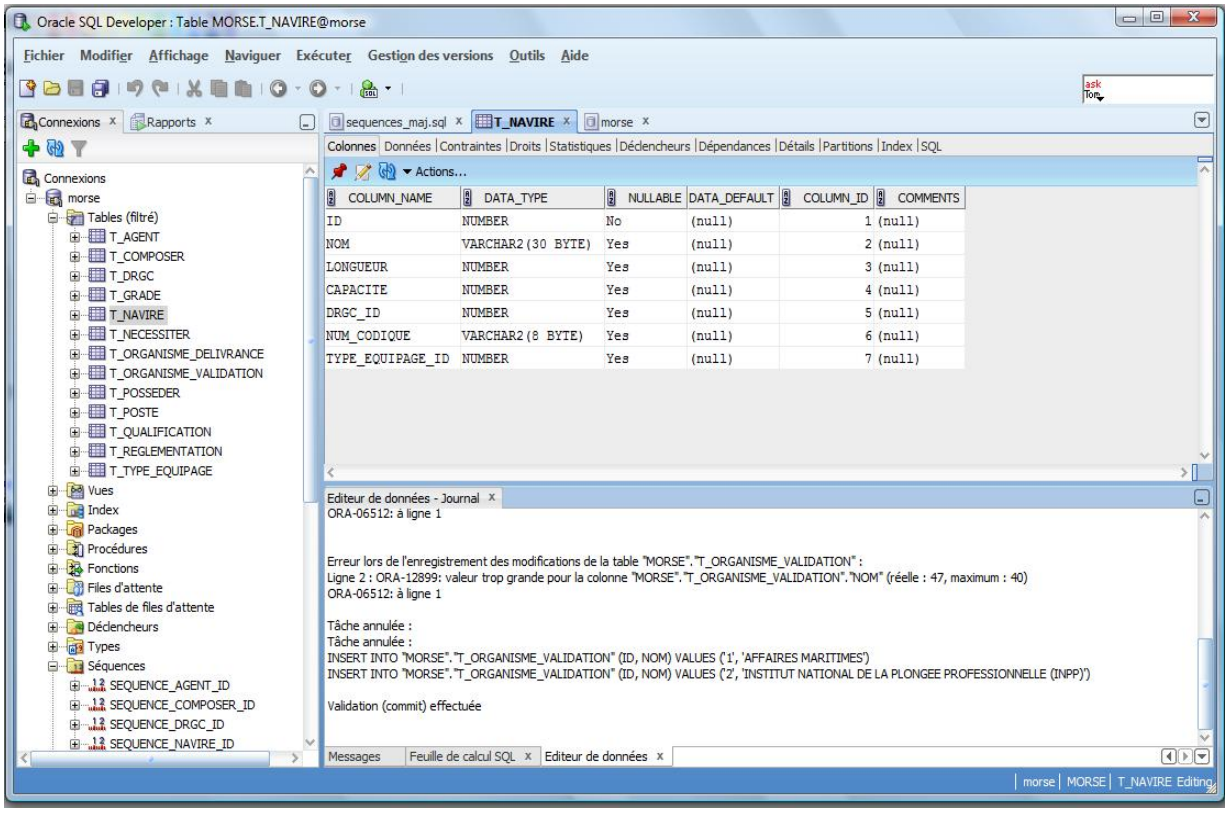

Oracle SQL Developer
Fichier Modifier Affichage Naviguer Exécuter Gestion des versions Outils Aide
Connexions
Rapports
sequences_maj.sql
T_ORGANISME_DELIVRANCE
morse
Colonnes Données Contraintes Droits Statistiques Déclencheurs Dépendances Détails Partitions Index SQL
Trier... Filtre :
Actions...
ID NOM
1 1 MARINE NATIONALE
2 2 MARINE MARCHANDE
3 3 AFFAIRES MARITIMES
Connexions
morse
Tables (filtré)
T_AGENT
T_COMPOSER
T_DRGC
T_GRADE
T_NAVIRE
T_NECESSITER
T_ORGANISME_DELIVRANCE
T_ORGANISME_VALIDATION
T_POSSEDER
T_POSTE
T_QUALIFICATION
T_REGLEMENTATION
T_TYPE_EQUIPAGE
Vues
Index
Packages
Procédures
Fonctions
Files d'attente
Tables de files d'attente
Déclencheurs
Types
Séquences
SEQUENCE_AGENT_ID
SEQUENCE_COMPOSER_ID
SEQUENCE_DRGC_ID
SEQUENCE_NAVIRE_ID
Editeur de données - Journal
ORA-06512: à ligne 1
Erreur lors de l'enregistrement des modifications de la table "MORSE"."T_ORGANISME_VALIDATION" :
Ligne 2 : ORA-12899: valeur trop grande pour la colonne "MORSE"."T_ORGANISME_VALIDATION"."NOM" (réelle : 47, maximum : 40)
ORA-06512: à ligne 1
Tâche annulée :
Tâche annulée :
INSERT INTO "MORSE"."T_ORGANISME_VALIDATION" (ID, NOM) VALUES ('1', 'AFFAIRES MARITIMES')
INSERT INTO "MORSE"."T_ORGANISME_VALIDATION" (ID, NOM) VALUES ('2', 'INSTITUT NATIONAL DE LA PLONGEE PROFESSIONNELLE (INPP)')
Validation (commit) effectuée
Messages
Feuille de calcul SQL
Editeur de données
Table MORSE.T_ORGANISME_DELIVRANCE@morse
Editing

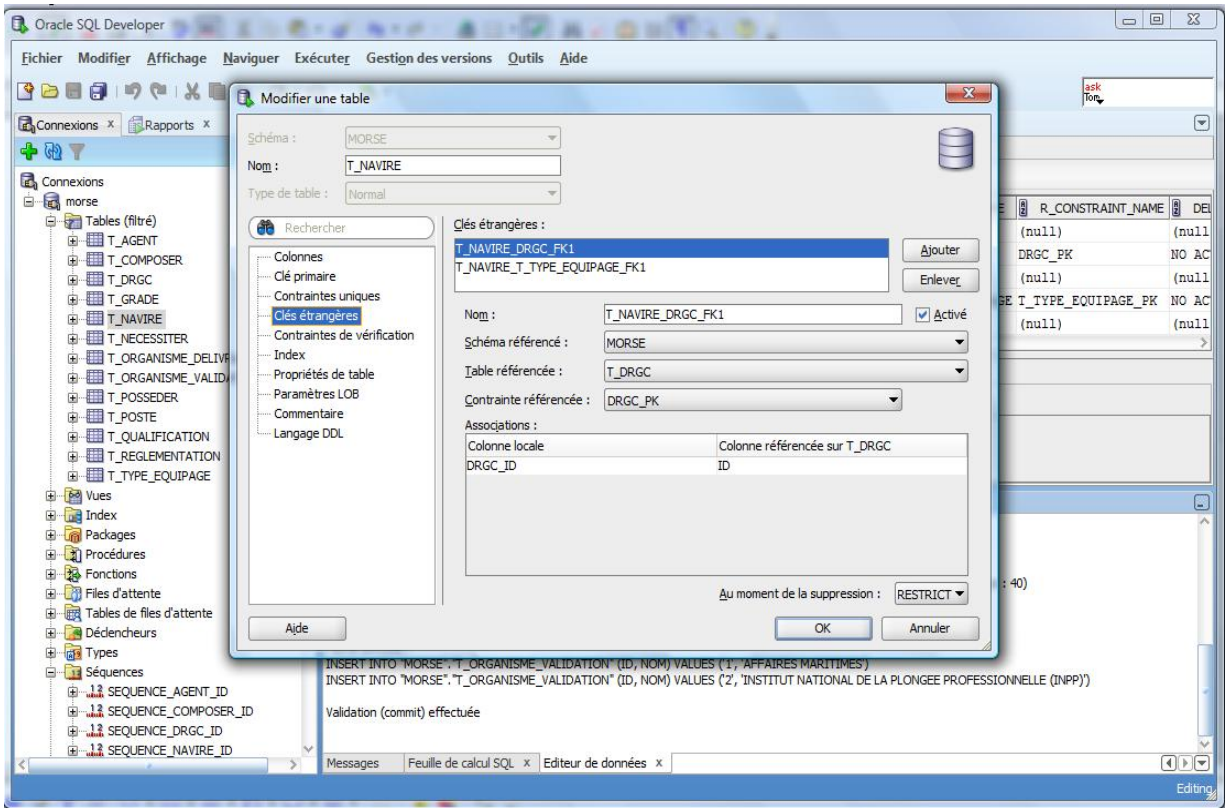
Un exemple de contrainte et de clés étrangères est donné ci-dessous.

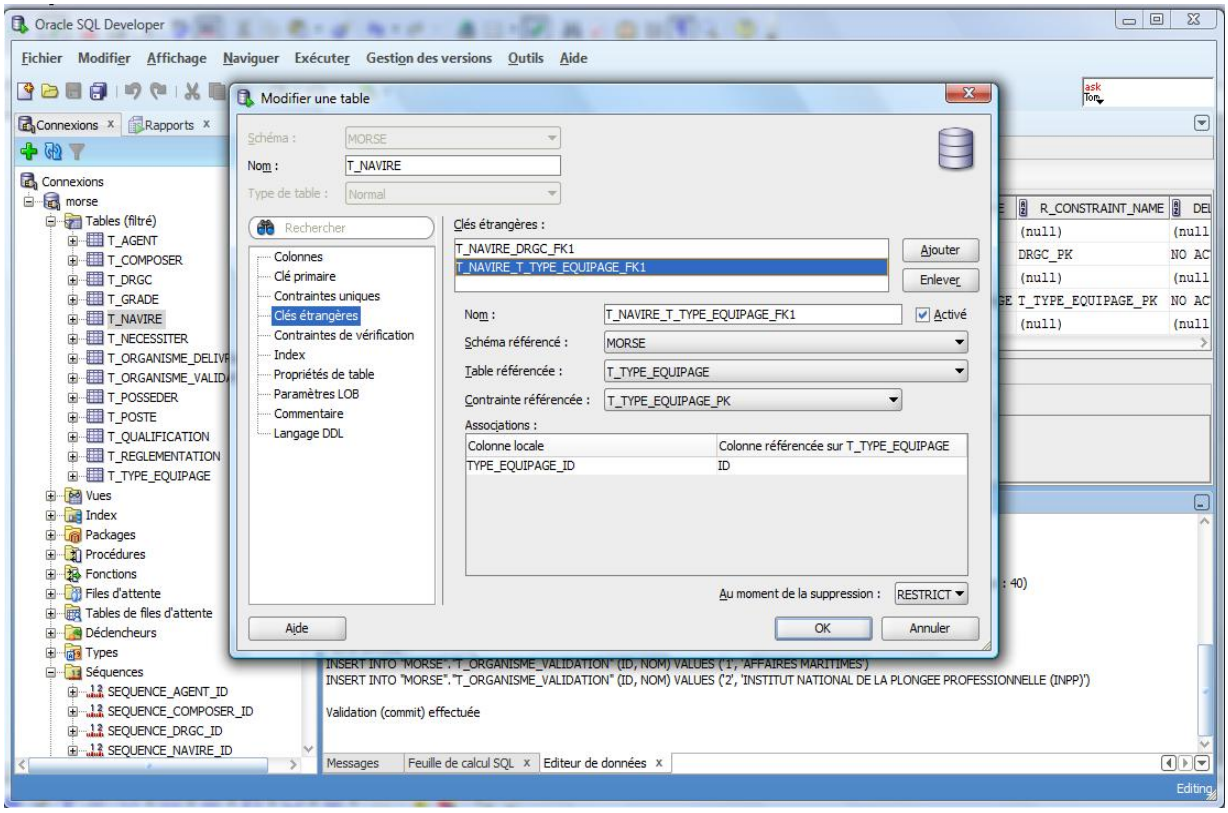
La clé primaire est sur le champs ID.

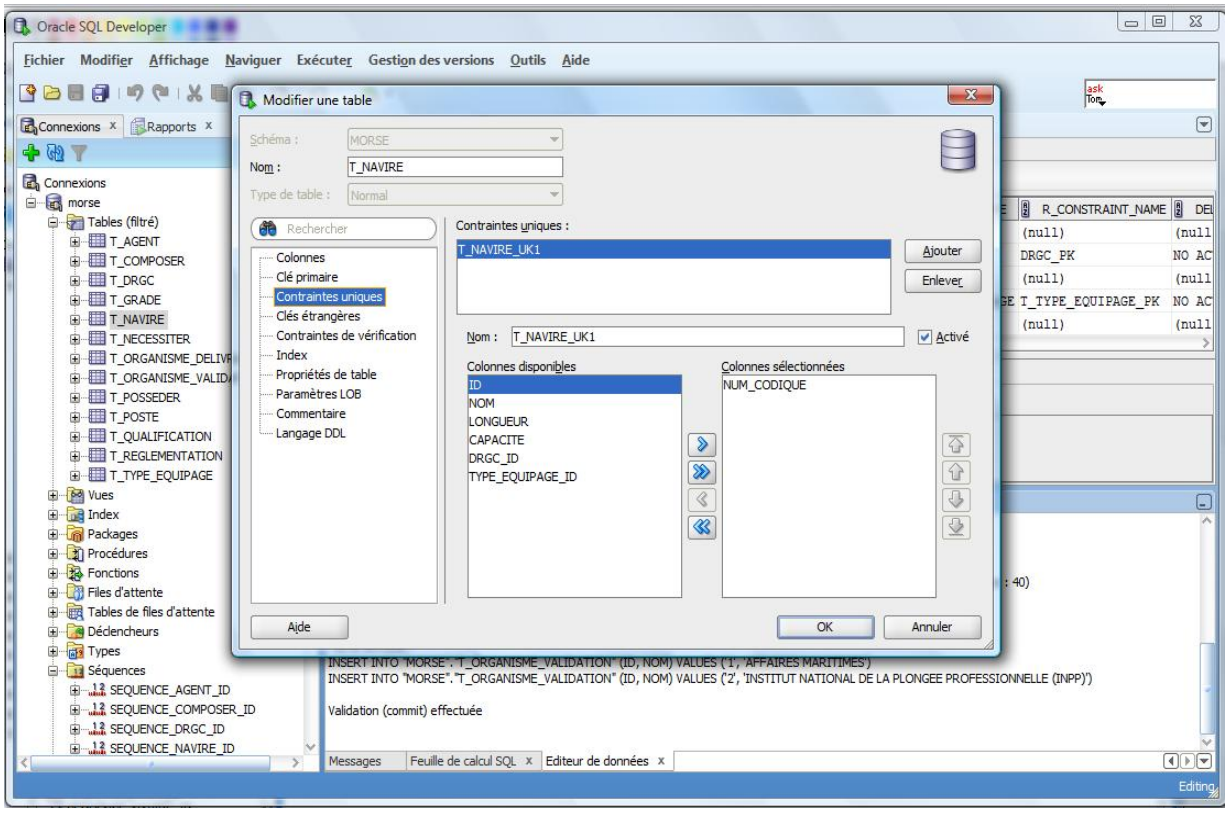
Les clés étrangères sont sur les champs DRGC_ID et TYPE_EQUIPAGE_ID.

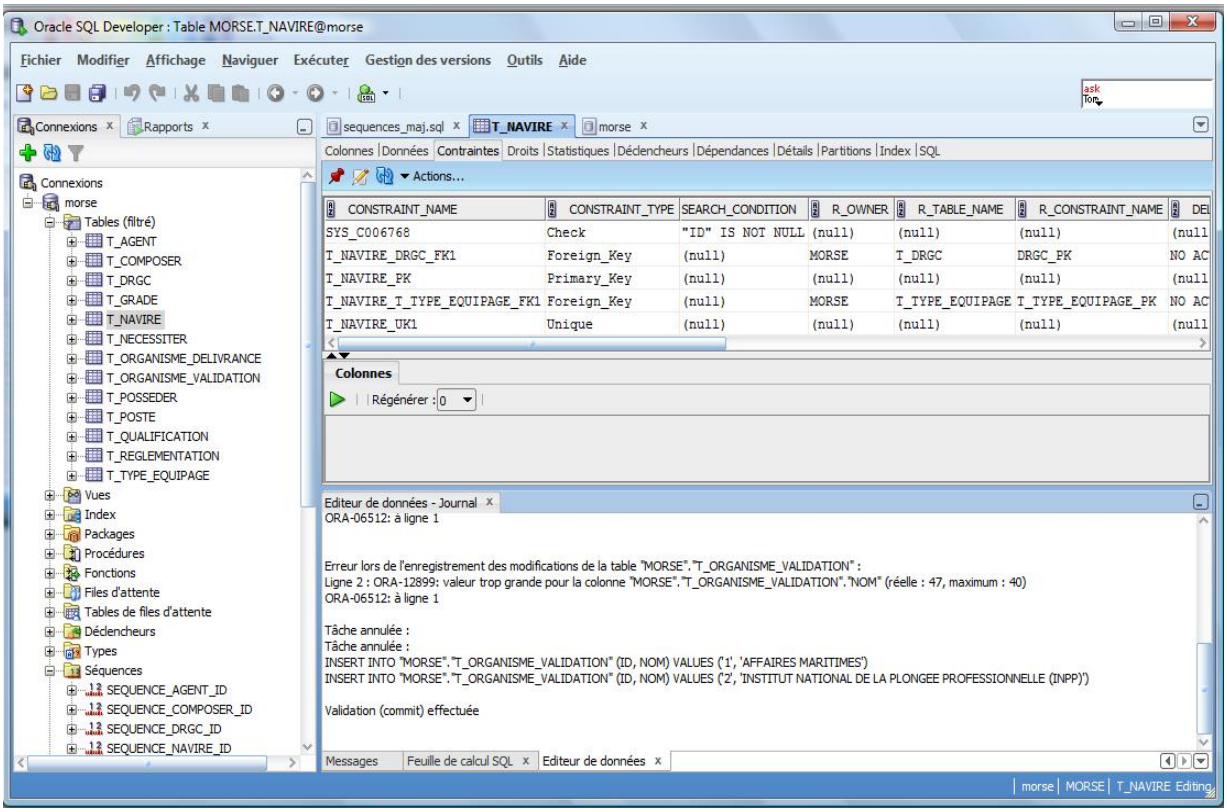
La contrainte «unique» est définie sur le champs NUM_CODIQUE.





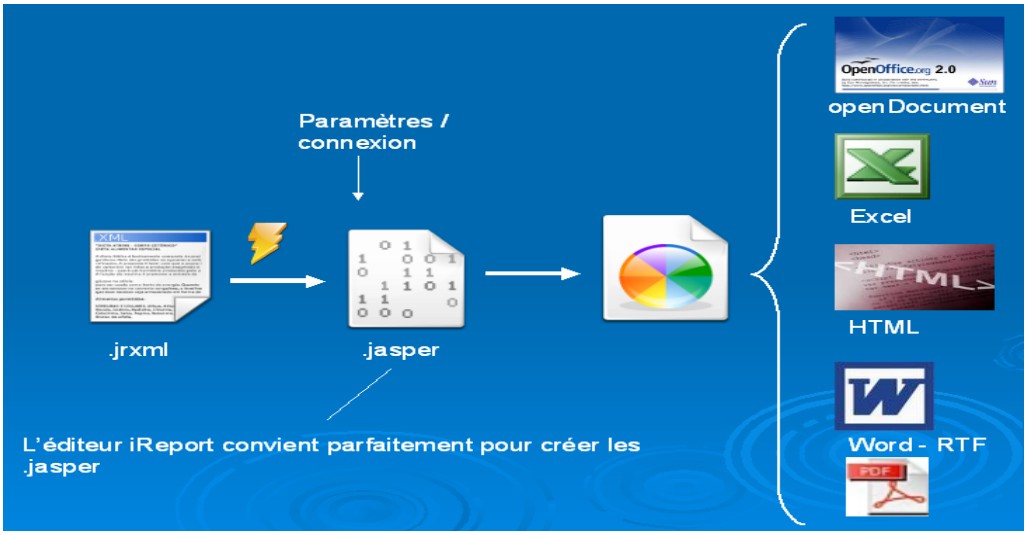

Oracle SQL Developer
Fichier Modifier Affichage Naviguer Exécuter Gestion des versions Outils Aide
Modifier une table
Schéma : MORSE
Nom : T_NAVIRE
Type de table : Normal
Rechercher
Colonnes
Clé primaire
Contraintes uniques
Clés étrangères
Contraintes de vérification
Index
Propriétés de table
Paramètres LOB
Commentaire
Langage DDL
Clés étrangères :
T_NAVIRE_DRGC_FK1
T_NAVIRE_T_TYPE_EQUIPAGE_FK1
Ajouter
Enlever
Nom : T_NAVIRE_T_TYPE_EQUIPAGE_FK1
Activé
Schéma référencé : MORSE
Table référencée : T_TYPE_EQUIPAGE
Contrainte référencée : T_TYPE_EQUIPAGE_PK
Associations :
Colonne locale
Colonne référencée sur T_TYPE_EQUIPAGE
TYPE_EQUIPAGE_ID
ID
Au moment de la suppression : RESTRICT
Aide
OK
Annuler
Connexions
Rapports
morse
Tables (filtré)
T_AGENT
T_COMPOSER
T_DRGC
T_GRADE
T_NAVIRE
T_NECESSITER
T_POSSEDER
T_POSTE
T_QUALIFICATION
T_REGLEMENTATION
T_TYPE_EQUIPAGE
Vues
Index
Packages
Procédures
Fonctions
Files d'attente
Déclencheurs
Types
Séquences
SEQUENCE_AGENT_ID
SEQUENCE_COMPOSER_ID
SEQUENCE_DRGC_ID
SEQUENCE_NAVIRE_ID
INSERT INTO "MORSE"."T_ORGANISME_VALIDATION" (ID, NOM) VALUES ('1', 'AFFAIRES MARITIMES')
INSERT INTO "MORSE"."T_ORGANISME_VALIDATION" (ID, NOM) VALUES ('2', 'INSTITUT NATIONAL DE LA PLONGEE PROFESSIONNELLE (INPP)')
Validation (commit) effectuée
Messages
Feuille de calcul SQL
Editeur de données
Editing


Oracle SQL Developer
Fichier Modifier Affichage Naviguer Exécuter Gestion des versions Outils Aide
Modifier une table
Schéma : MORSE
Nom : T_NAVIRE
Type de table : Normal
Rechercher
Colonnes
Clé primaire
Contraintes uniques
Clés étrangères
Contraintes de vérification
Index
Propriétés de table
Paramètres LOB
Commentaire
Langage DDL
Contraintes uniques :
T_NAVIRE_UK1
Ajouter
Enlever
Nom : T_NAVIRE_UK1
Activé
Colonnes disponibles
ID
NOM
LONGUEUR
CAPACITE
DRGC_ID
TYPE_EQUIPAGE_ID
Colonnes sélectionnées
NUM_CODIQUE
Aide
OK
Annuler
Connexions
Rapports
morse
Tables (filtré)
T_AGENT
T_COMPOSER
T_DRGC
T_GRADE
T_NAVIRE
T_NECESSITER
T_POSSEDER
T_POSTE
T_QUALIFICATION
T_REGLEMENTATION
T_TYPE_EQUIPAGE
Vues
Index
Packages
Procédures
Fonctions
Files d'attente
Déclencheurs
Types
Séquences
SEQUENCE_AGENT_ID
SEQUENCE_COMPOSER_ID
SEQUENCE_DRGC_ID
SEQUENCE_NAVIRE_ID
INSERT INTO "MORSE"."T_ORGANISME_VALIDATION" (ID, NOM) VALUES ('1', 'AFFAIRES MARITIMES')
INSERT INTO "MORSE"."T_ORGANISME_VALIDATION" (ID, NOM) VALUES ('2', 'INSTITUT NATIONAL DE LA PLONGEE PROFESSIONNELLE (INPP)')
Validation (commit) effectuée
Messages
Feuille de calcul SQL
Editeur de données
Editing



3.3.4. Un outil de reporting : JasperReports

JasperReports est apparu au début des années 2000, c'est un outil de reporting qui peut restituer à l'écran, sur une imprimante ou vers un fichier PDF[53], HTML[54], Excel, ...



[53] PDF : Portable Document Format, format de fichier non modifiable et lisible quelque soit l'application ou la plateforme utilisée.

[54] HTML : HyperText Mark-up Language

Il peut être utilisé dans toute application JAVA (y compris Java EE) ou application web pour produire des états dynamiques. Il exécute des instructions d'un fichier XML [55] ou Jasper.

Il se présente sous forme de bibliothèque libre et ses caractéristiques sont :

● sa compatibilité avec plusieurs types de fichiers qui permet à travers les bibliothèques JfreeChart, d'inclure des graphiques de divers types,

● la possibilité de regrouper plusieurs sources. Des données peuvent être récupérées de sources identifiées telles que JDBC, XML, Hibernate et de valeurs séparées par des virgules

● des scriplets (désigne un type d'applet dans lequel on a substitué un script en Javascript au Java d'origine) qui peuvent être inclus dans l'état et appelé à tout niveau. Les scriplets sont écrits en JAVA et se déclinent au niveau de l'état, de la page, de la colonne ou du groupe

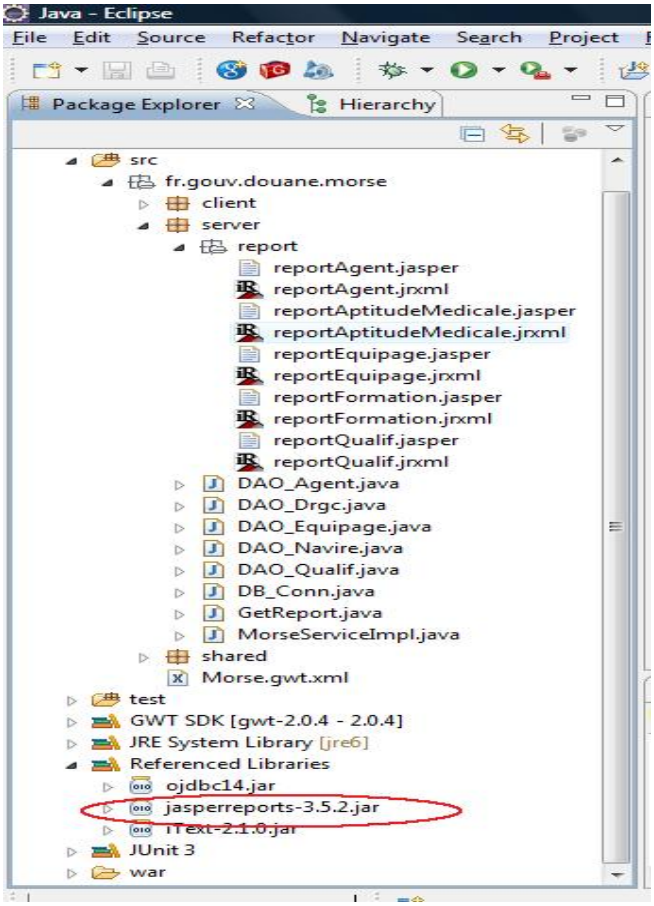
● de permettre la génération de sous-états

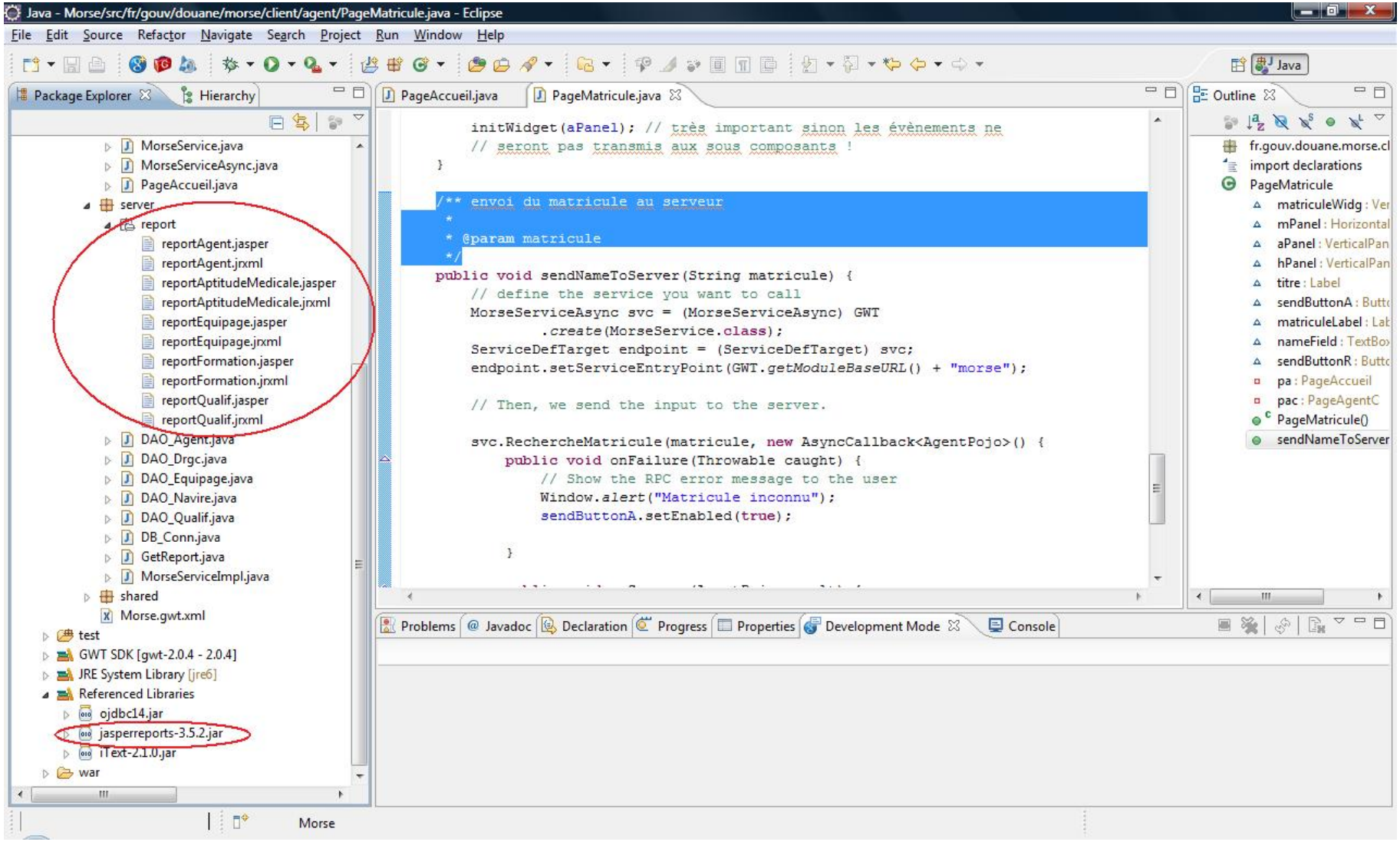
Pour les utilisateurs avertis qui ont des besoins plus complexes, les états issus de JasperReports peuvent être importés sur le JasperServer, un serveur d'états intéractif.

JasperReports a donc été mis en œuvre pour générer les états requis de l'application MORSE. Ces états ont été définis dans le CDCT.

Afin de l'intégrer à la plateforme Eclipse, il a fallu intégrer la bibliothèque concernée dans l'arborescence de l'application.

[55] XML : eXtensible Mark-up Language

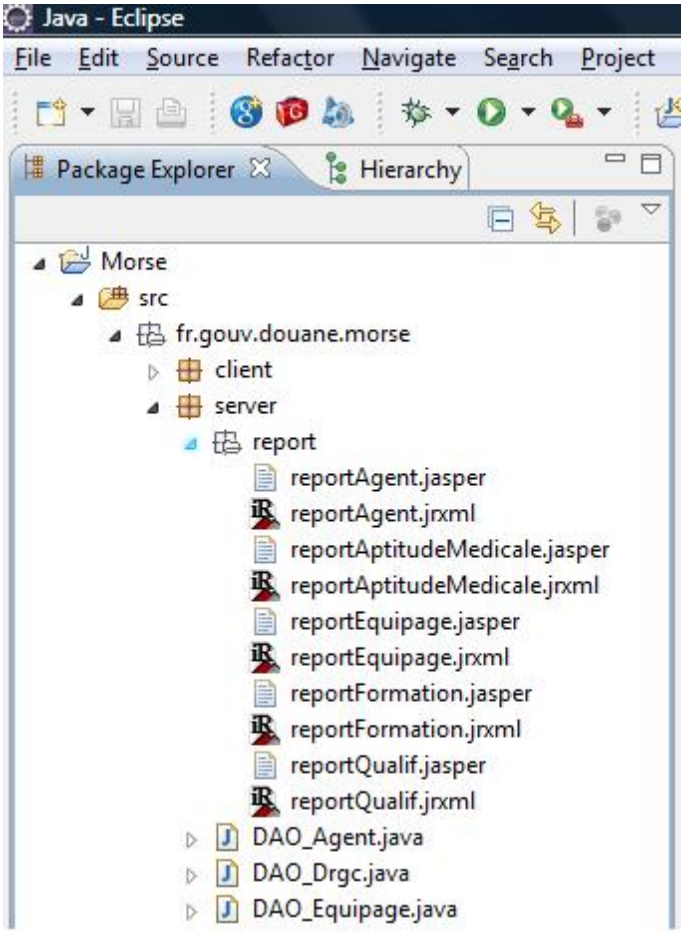

Java - Eclipse
File Edit Source Refactor Navigate Search Project
Package Explorer
Hierarchy
src
fr.gouv.douane.morse
client
server
report
reportAgent.jasper
reportAgent.jrxml
reportAptitudeMedicale.jasper
reportAptitudeMedicale.jrxml
reportEquipage.jasper
reportEquipage.jrxml
reportFormation.jasper
reportFormation.jrxml
reportQualif.jasper
reportQualif.jrxml
DAO_Agent.java
DAO_Drgc.java
DAO_Equipage.java
DAO_Navire.java
DAO_Qualif.java
DB_Conn.java
GetReport.java
MorseServiceImpl.java
shared
Morse.gwt.xml
test
GWT SDK [gwt-2.0.4 - 2.0.4]
JRE System Library [jre6]
Referenced Libraries
ojdbc14.jar
jasperreports-3.5.2.jar
iText-2.1.0.jar
JUnit 3
war


Java - Morse/src/fr/gouv/douane/morse/client/agent/PageMatricule.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
Hierarchy
MorseService.java
MorseServiceAsync.java
PageAccueil.java
server
report
reportAgent.jasper
reportAgent.jrxml
reportAptitudeMedicale.jasper
reportAptitudeMedicale.jrxml
reportEquipage.jasper
reportEquipage.jrxml
reportFormation.jasper
reportFormation.jrxml
reportQualif.jasper
reportQualif.jrxml
DAO_Agent.java
DAO_Drgc.java
DAO_Equipage.java
DAO_Navire.java
DAO_Qualif.java
DB_Conn.java
GetReport.java
MorseServiceImpl.java
shared
Morse.gwt.xml
test
GWT SDK [gwt-2.0.4 - 2.0.4]
JRE System Library [jre6]
Referenced Libraries
ojdbc14.jar
jasperreports-3.5.2.jar
iText-2.1.0.jar
war
PageAccueil.java
PageMatricule.java
initWidget(aPanel); // très important sinon les évènements ne
// seront pas transmis aux sous composants !
}
/** envoi du matricule au serveur
*
* @param matricule
*/
public void sendNameToServer(String matricule) {
// define the service you want to call
MorseServiceAsync svc = (MorseServiceAsync) GWT
.create(MorseService.class);
ServiceDefTarget endpoint = (ServiceDefTarget) svc;
endpoint.setServiceEntryPoint(GWT.getModuleBaseURL() + "morse");
// Then, we send the input to the server.
svc.RechercheMatricule(matricule, new AsyncCallback<AgentPojo>() {
public void onFailure(Throwable caught) {
// Show the RPC error message to the user
Window.alert("Matricule inconnu");
sendButtonA.setEnabled(true);
}
Outline
fr.gouv.douane.morse.cl
import declarations
PageMatricule
matriculeWidg : Ver
mPanel : Horizontal
aPanel : VerticalPan
hPanel : VerticalPan
titre : Label
sendButtonA : Butt
matriculeLabel : Lab
nameField : TextBox
sendButtonR : Butt
pa : PageAccueil
pac : PageAgentC
PageMatricule()
sendNameToServer
Problems
Javadoc
Declaration
Progress
Properties
Development Mode
Console
Morse

Pour constituer les états, j'ai utilisé l'éditeur iReport, logiciel de type WYSIWYG (What You See Is What You Get) qui permet de créer des états pour JasperReport.

Les états sont de type « jrxml » et une fois compilés de type « jasper ».

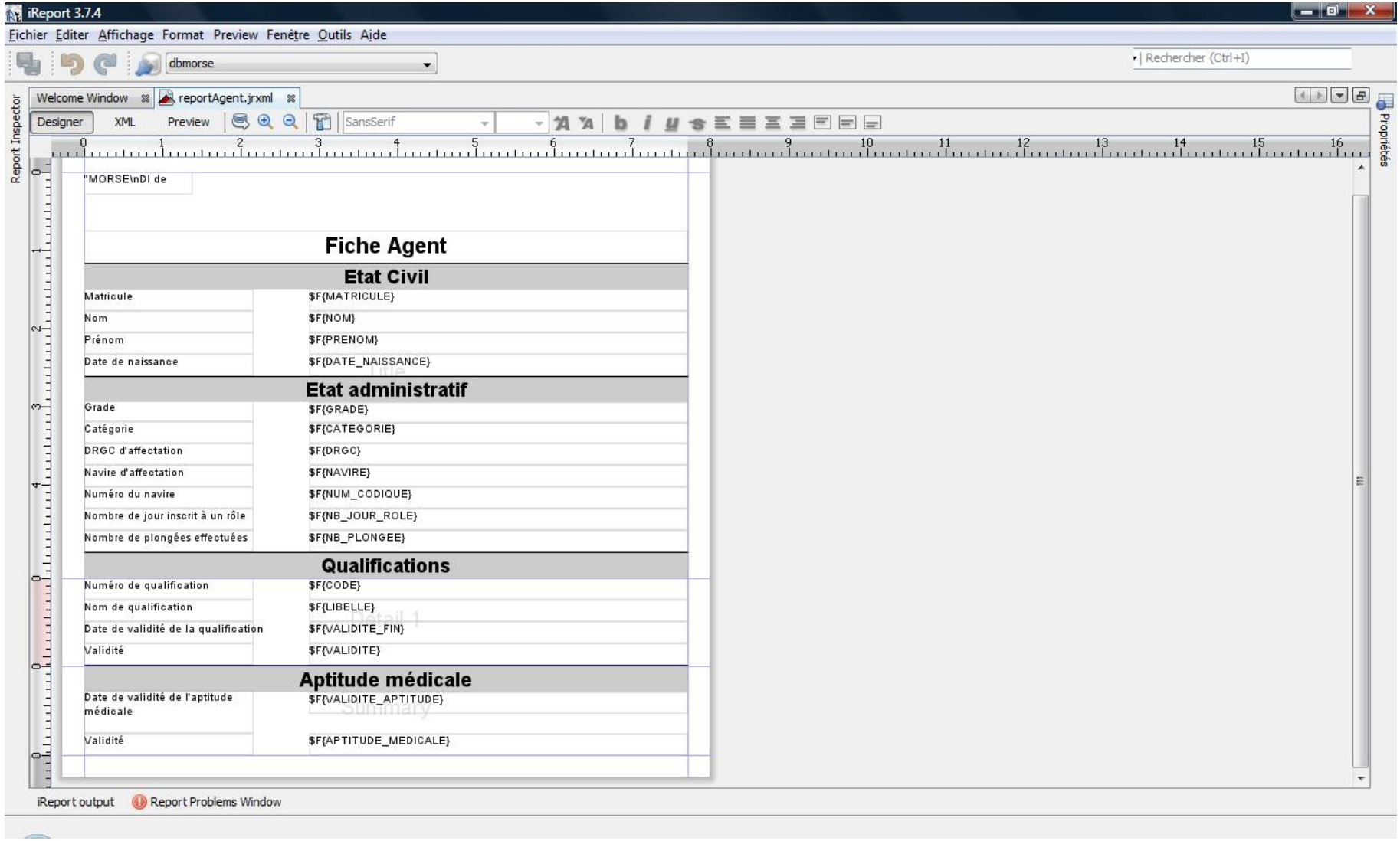
Ils sont repris dans le répertoire « serveur » de l'application MORSE comme ci-dessous :



Les états de MORSE sont élaborés à partir de certaines données de la base de données Oracle 10g. Ainsi, pour constituer la fiche Agent, les données sont issues des tables T_AGENT, T_GRADE, T_DRGC, T_NAVIRE, T_QUALIFICATION.

L'écran iReport ci-dessous reprend ces données et montre la mise en page qui en a été faite.

Les champs issus de la base de données sont représentés comme *$F{MATRICULE}*.


iReport 3.7.4
Fichier Editer Affichage Format Preview Fenêtre Outils Aide
dbmorse
Rechercher (Ctrl+I)
Welcome Window
reportAgent.jrxml
Designer
XML
Preview
SansSerif
Report Inspector
Propriétés
"MORSE\nDI de
Fiche Agent
Etat Civil
Matricule
$F{MATRICULE}
Nom
$F{NOM}
Prénom
$F{PRENOM}
Date de naissance
$F{DATE_NAISSANCE}
Etat administratif
Grade
$F{GRADE}
Catégorie
$F{CATEGORIE}
DRGC d'affectation
$F{DRGC}
Navire d'affectation
$F{NAVIRE}
Numéro du navire
$F{NUM_CODIQUE}
Nombre de jour inscrit à un rôle
$F{NB_JOUR_ROLE}
Nombre de plongées effectuées
$F{NB_PLONGEE}
Qualifications
Numéro de qualification
$F{CODE}
Nom de qualification
$F{LIBELLE}
Date de validité de la qualification
$F{VALIDITE_FIN}
Validité
$F{VALIDITE}
Aptitude médicale
Date de validité de l'aptitude médicale
$F{VALIDITE_APTITUDE}
Validité
$F{APTITUDE_MEDICALE}
iReport output
Report Problems Window

Le résultat obtenu sous Eclipse est le suivant :

MORSE
DI de ROUEN
DRGC MANCHE
MER DU NORD

Fiche Agent

Etat Civil

Matricule	12345
Nom	DUPONT
Prénom	JEAN
Date de naissance	12/12/1960

Etat administratif

Grade	1
Catégorie	A
DRGC d'affectation	MANCHE MER DU NORD
Navire d'affectation	VENT D'AMONT
Numéro du navire	DF 1
Nombre de jour inscrit à un rôle	12
Nombre de plongées effectuées	1

Qualifications

Numéro de qualification	null
Nom de qualification	null
Date de validité de la qualification	null
Validité	N

Aptitude médicale

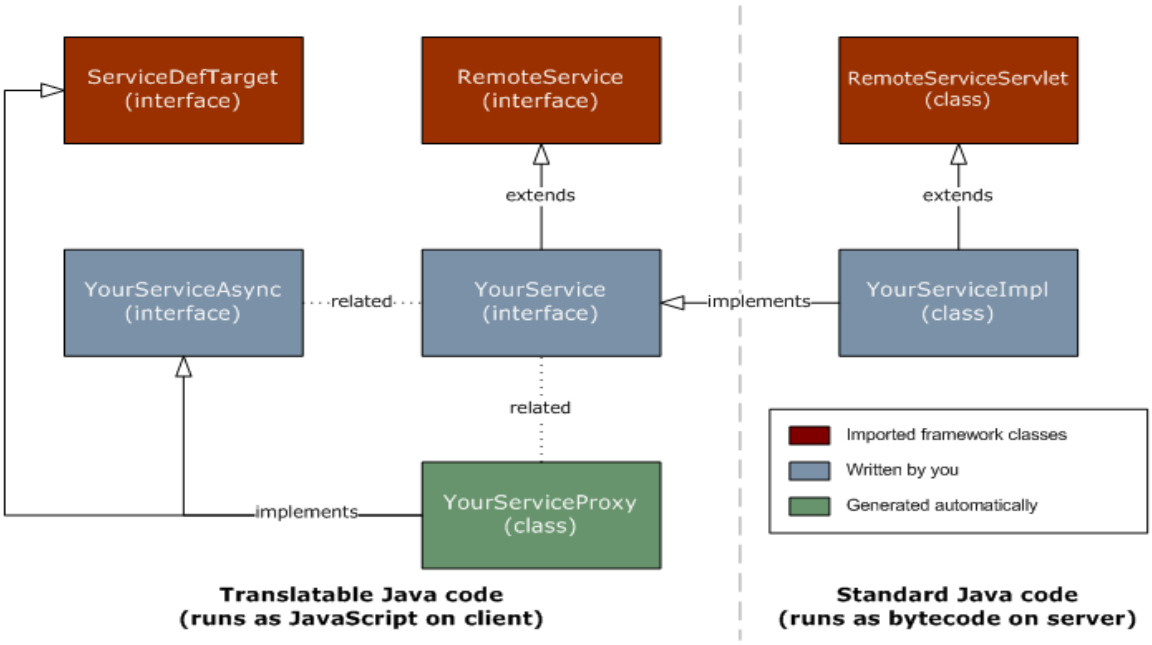
Date de validité de l'aptitude médicale	12/01/2011
Validité	O

L'éditeur iReports permet tout comme l'AGL Eclipse de se « prémunir » d'un codage fastidieux lié aux mises en page.

De plus, la création d'état n'est pas restreinte et la MOA peut requérir de nouveaux états selon l'évolution de ses besoins.

3.3.5. Mise en œuvre d'une connexion RPC

L'acronyme RPC désigne un protocole qui permet de faire des appels au serveur à partir d'un ordinateur distant. Il est souvent utilisé dans un système client-serveur et gère les messages échangés entre ces deux entités.



Pour créer une interface RPC, il faut :

- créer une interface (YourService) qui hérite du service « RemoteService » et décrit toutes les méthodes RPC utilisées
- créer une classe (YourServiceImpl) qui définit le code du côté serveur et qui hérite du « RemoteServiceServlet » tout en implémentant l'interface créée ci-dessus. Ainsi la classe « YourServiceImpl » (qui contient les déclarations des méthodes) garantit que ces méthodes seront appliquées dans l'interface qu'elle implémente (YourService)
- créer une interface asynchrone (YourServiceAsync) qui sera appelée par le code « client ».

Ainsi, sur le schéma ci-dessus, on observe une partie « client » à gauche et une partie « serveur » à droite. La classe « YourServiceProxy » est générée automatiquement, la classe « YourServiceImpl » et les interfaces « YourService » et « YourServiceAsync »

sont écrites par le développeur, la classe « RemoteServiceServlet » et les interfaces « RemoteService » et « ServiceDefTarget » sont importées des classes natives.

L'interface synchrone (YourService) définit les spécifications du service « client ». Chaque classe du côté « serveur » (YourServiceImpl) doit hériter de la classe native « RemoteServiceServlet » et implémenter cette interface synchrone.
Il est impossible d'adresser directement à partir du « client » cette interface synchrone. Il faut pour cela créer une interface asynchrone comme ci-dessous.

L'interface asynchrone qui est implémentée par la classe « YourServiceProxy » est en relation avec l'interface synchrone. Lors d'un appel au serveur, un objet de « callback » est adressé pour permettre d'être notifié de la fin de l'appel asynchrone, durant ce laps de temps, l'appelant ne peut pas être bloqué. Une méthode asynchrone ne renvoie rien (void).

L'implémentation d'un service sert à répondre aux requêtes du « client ». Ce service doit hériter de la classe « RemoteServiceServlet » et doit implémenter l'interface service associé.

Pour mettre en œuvre une connexion RPC dans l'application MORSE, toutes ces interfaces ont été créées, elles sont décrites dans le paragraphe 3.3.2.2 ci-dessus.
L'interface «YourServiceImpl» s'appelle «MorseServiceImpl», «YourService» s'appelle «MorseService» et «YourServiceAsync» s'appelle «MorseServiceAsync».

3.3.6. Hibernate

3.3.6.1. Présentation de Hibernate

La programmation objet impose la manipulation d'objets. Une base de données relationnelle n'est pas perçue comme un objet. De ce fait, il faut transformer ces données en objet.
Dans l'application MORSE, ceci a été fait à l'aide des POJO et DAO, cependant, la programmation a été plus fastidieuse car il a été nécessaire de décrire chacune des données pour la rendre «objet».

De plus, ces objets ne sont pas persistants et contraignent l'application à les recréer à chaque démarrage.

La persistance se réfère au mécanisme de sauvegarde et de restauration de données, pour qu'un programme puisse se terminer sans que ses données ni son état d'exécution soient perdus.

Hibernate est un framework gratuit, sous licence GNU Lesser General Public license, qui gère les données persistantes en remplaçant les accès à la base de données relationnelle par des appels à des méthodes objet de haut niveau.

C'est un outil de «Object/Relational Mapping (ORM)» pour des environments JAVA. Le terme ORM se rapporte à la technique de programmation qui permet de donner l'illusion qu'une base de données relationnelle est une base de données orientée objet, en créant des correspondances.

Hibernate prend en charge le «mapping» des classes JAVA vers les tables de la base de données (et des données JAVA vers des données de type SQL) et permet aussi d'effectuer des requêtes.

Il peut significativement réduire le temps de développement consacré à la définition des bases de données (SQL et JDBC[56]) en objets.

3.3.6.2. MORSE et Hibernate

La programmation de l'application MORSE n'a pas été faite à l'aide de Hibernate. À la mise en place de la plateforme de développement, j'ai installé Eclipse et GWT[57] (outils de développement web). Pour permettre la gestion des objets persistants de la base de données, j'ai tenté à maintes reprises d'installer Hibernate sur ma plateforme Eclipse mais pour une raison que je n'ai pas réussi à analyser, l'installation rendait l'AGL inutilisable.

Par manque de temps, il m'était devenu impératif d'avancer sur la programmation de l'application et je décidais de programmer en «manuel» les objets de la base de données à travers les POJO et les DAO.

Cependant, cette solution ne sera que temporaire, le temps que l'application soit testée en local à la DRGC MMN.

Si cette application devait être généralisée à l'ensemble des DRGC, il sera nécessaire de prévoir l'installation de Hibernate et le mapping des objets persistants.

[56] JDBC : Java DataBase Connectivity

[57] GWT : Google Web Tool kit

De plus, l'utilisation d'Hibernate facilite la maintenance de l'application car si une table de la base de données devait être modifiée, toute la programmation en relation avec cette table devra être mise à jour. Ceci peut engendrer une perte de temps significative.

Hibernate est repris au palier technique de la DGDDI et son utilisation respectera les normes imposées par l'administration.

4. LE RESULTAT : L'APPLICATION MORSE

4.1. Le déploiement

Pour permettre le déploiement d'une application, GWT propose plusieurs solutions. Il peut se faire :

- pour un serveur web : il suffit de copier les fichiers de l'application générés par GWT après compilation et de les héberger sur le serveur web. Il faudra également modifier le code côté «serveur» de l'application pour permettre la communication entre le serveur et le client. Cette communication pourra être de type JSONP (JavaScript Object Notation with Padding), de type scripts serveur capable de traiter les requêtes sous HTTP à travers la classe RequestBuilder de GWT ou encore de type RPC.
- pour un conteneur de servlet utilisant une connexion RPC : le déploiement se fait facilement grâce au compilateur GWT qui génère les fichiers dans un répertoire ayant une structure compatible avec les spécifications de l'API Servlet 2.5.
- pour le Google App Engine : il est nécessaire de compiler l'application pour générer les fichiers dans la structure standard du répertoire «war», puis l'application peut être déployée à l'aide de l'utilitaire appcfg.

L'application MORSE sera mise à disposition de la DRGC-MMN début octobre 2010, la phase de développement ayant été rallongée. La livraison initiale avait été prévue pour le 09 novembre 2009.
L'application sera installée en premier lieu à la DI de Rouen où des agents désignés procèderont tout d'abord à l'alimentation de la base de données puis effectueront les tests fonctionnels.
À ce stade, l'application ne sera pas déployée par l'une des méthodes décrite ci-dessus.

Tous les téléservices de la DGDDI sont régis par des règles de sécurité inhérentes à l'intranet Aladin.
Si l'application MORSE devait être installée pour tests sur un poste relié à l'intranet, elle pourrait constituer une faille de sécurité de par son accès à une base de données.

Dans l'état actuel de l'application, l'authentification des utilisateurs par API RUSH n'a pas été implémentée.

L'installation d'une base de données est complexe et engendre l'installation de plusieurs services et applicatifs par défaut (machine virtuelle java, serveur web, ...).

Plusieurs failles classiques sont identifiées sur les bases de données, buffer overflows, bugs d'authentification, injections SQL.

Mais il existe également des failles de sécurité liées aux applications associées (serveurs web d'administration, applications web, programme setuid, ...) et aux mauvaises configurations.

L'injection SQL sur une application permet d'introduire une requête SQL non prévue par l'application et de compromettre la sécurité de la base de données.

De plus, le système d'exploitation peut être « attaqué » à travers la base de données dont les accès seraient mal configurés.

Le listener qui est un élément logiciel à l'écoute des évènements sur le réseau pour exécuter les tâches requises, peut être accédé à distance pour obtenir des informations sur la base de données et peut même être arrêté à distance.

Pour ces raisons, j'ai décidé d'installer MORSE sur un PC portable fourni par la DI de ROUEN. Les futurs utilisateurs pourront mener les tests prévus en autonome et sans risques pour l'intranet douanier.

Dans l'environnement normal des centres de calcul, j'aurais eu accès à un serveur de validation qui m'aurait permis de déployer l'application selon l'une des méthodes de déploiement décrites ci-dessus, dans un environnement semblable à celui de production.

De plus, l'utilisation d'un PC portable permettra de rendre ce poste de tests « mobile » pour les utilisateurs géographiquement répartis, ce qui est le cas pour les agents de la DI de Rouen et ceux de la DRGC MMN.

Ce poste fera office de client et de serveur tout comme en environnement de développement sur mon propre PC.

Le déploiement de l'application ne pourra s'effectuer qu'après le recettage de l'application par la MOA. La méthode de déploiement sera choisie par le CID, selon les normes appliquées dans ce centre et dans le respect du palier technique de la DGDDI.

4.2. Les tests

Les tests unitaires sont des tests qui permettent de vérifier que l'application fonctionne correctement et conformément aux besoins définis. Ils sont l'apanage du développeur. Chaque test est défini pour un module ou une fonctionnalité en ayant une entrée et une sortie (résultat attendu).

Le test peut être simple, comme «cliquez sur le bouton *Retour*» et vérifier que l'on revient bien à la page précédente mais peut aussi être plus complexe, enchainant une série d'actions pour tester une fonctionnalité traduisant le processus métier à automatiser.

Le développeur constitue un jeu de tests qui pourra être rejoué autant que nécessaire afin de tester l'application en phase de développement mais aussi en phase de maintenance en cas de modification de bug ou de modification évolutive.

L'utilisation d'Eclipse permet de générer des tests unitaires automatiquement à l'aide d'une bibliothèque JUNIT pour le langage de programmation Java.

JUnit définit deux types de fichiers de tests :

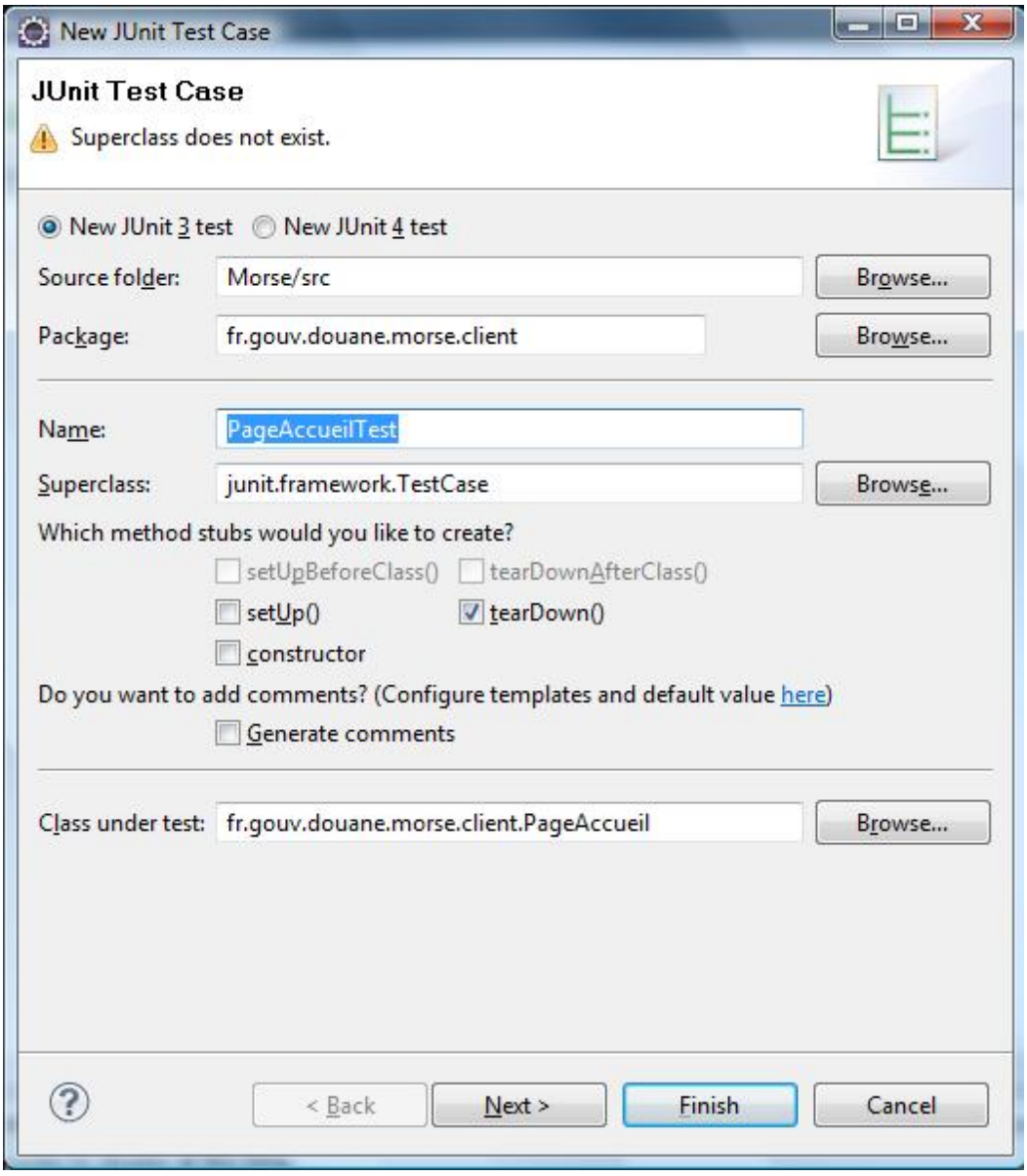
●les TestCase qui contiennent un certain nombre de méthodes de tests et qui servent à tester le bon fonctionnement d'une classe.

●Une TestSuite qui permet d'exécuter un certain nombre de TestCase pré-définis.

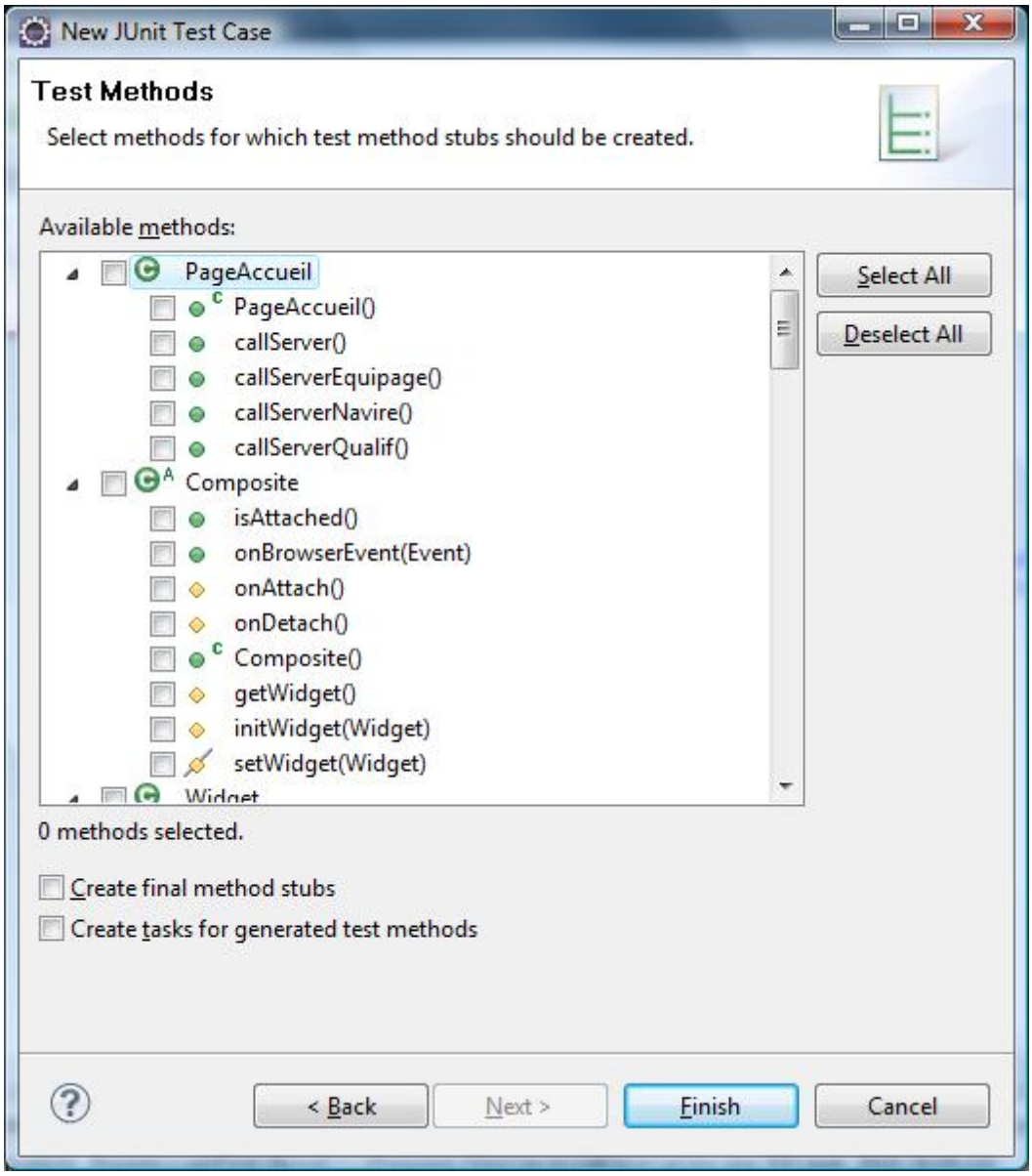
Il permet de tester que le fonctionnement d'une application est correct, on pourrait dire qu'il remplace les fameux « println » de débuggage utilisés par la plupart des programmeurs et peut même aller plus loin et rejoindre les principes de « l'extrême programming » qui en compte 12.

Les écrans ci-dessous donnent un exemple de création d'un TestCase avec l'AGL Eclipse.

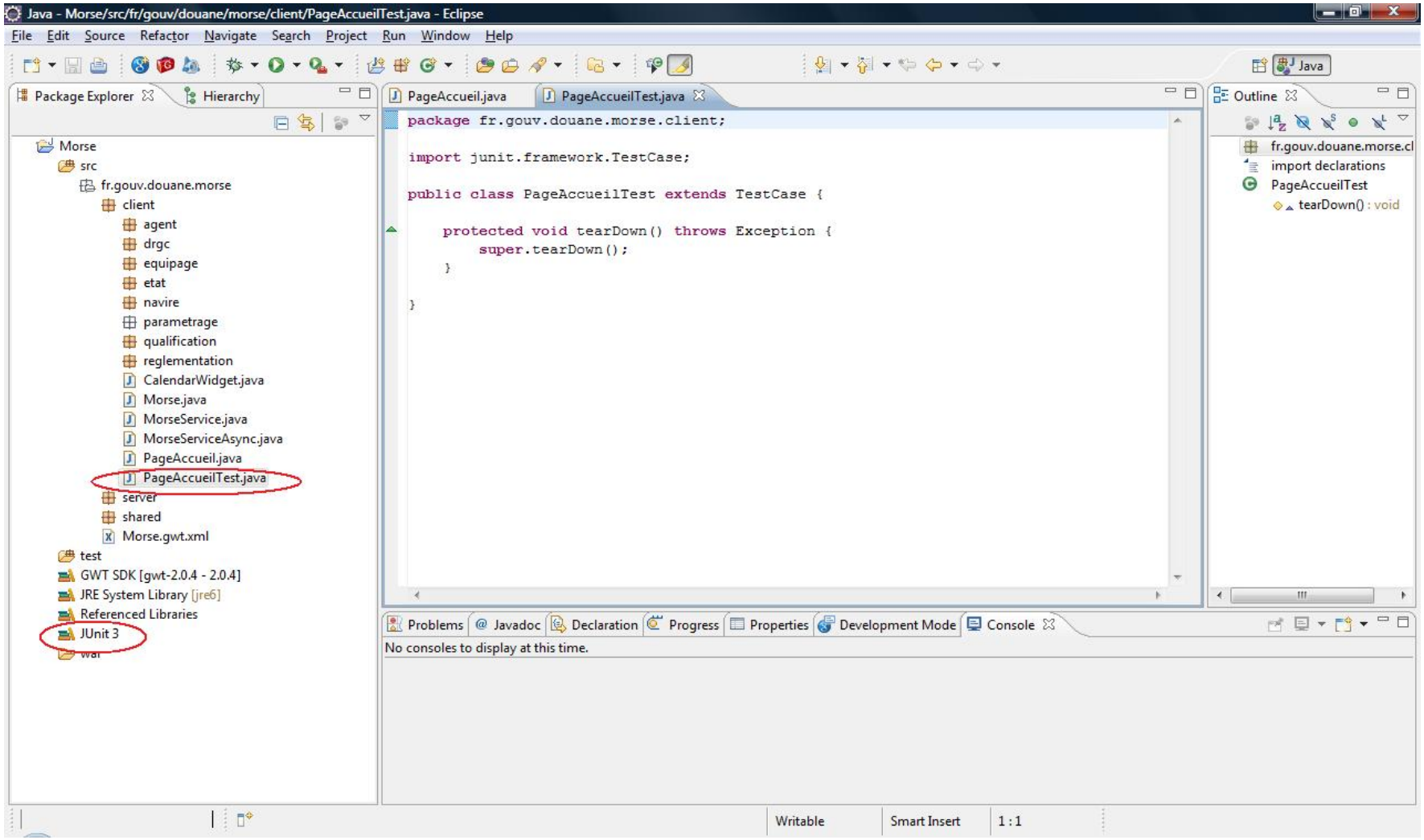
Pour créer un TestCase, on utilise le menu « File/New/JUnit TestCase, on obtient l'écran :


New JUnit Test Case
JUnit Test Case
Superclass does not exist.
New JUnit 3 test
New JUnit 4 test
Source folder:
Morse/src
Browse...
Package:
fr.gouv.douane.morse.client
Browse...
Name:
PageAccueilTest
Superclass:
junit.framework.TestCase
Browse...
Which method stubs would you like to create?
setUpBeforeClass()
tearDownAfterClass()
setUp()
tearDown()
constructor
Do you want to add comments? (Configure templates and default value here)
Generate comments
Class under test:
fr.gouv.douane.morse.client.PageAccueil
Browse...
< Back
Next >
Finish
Cancel

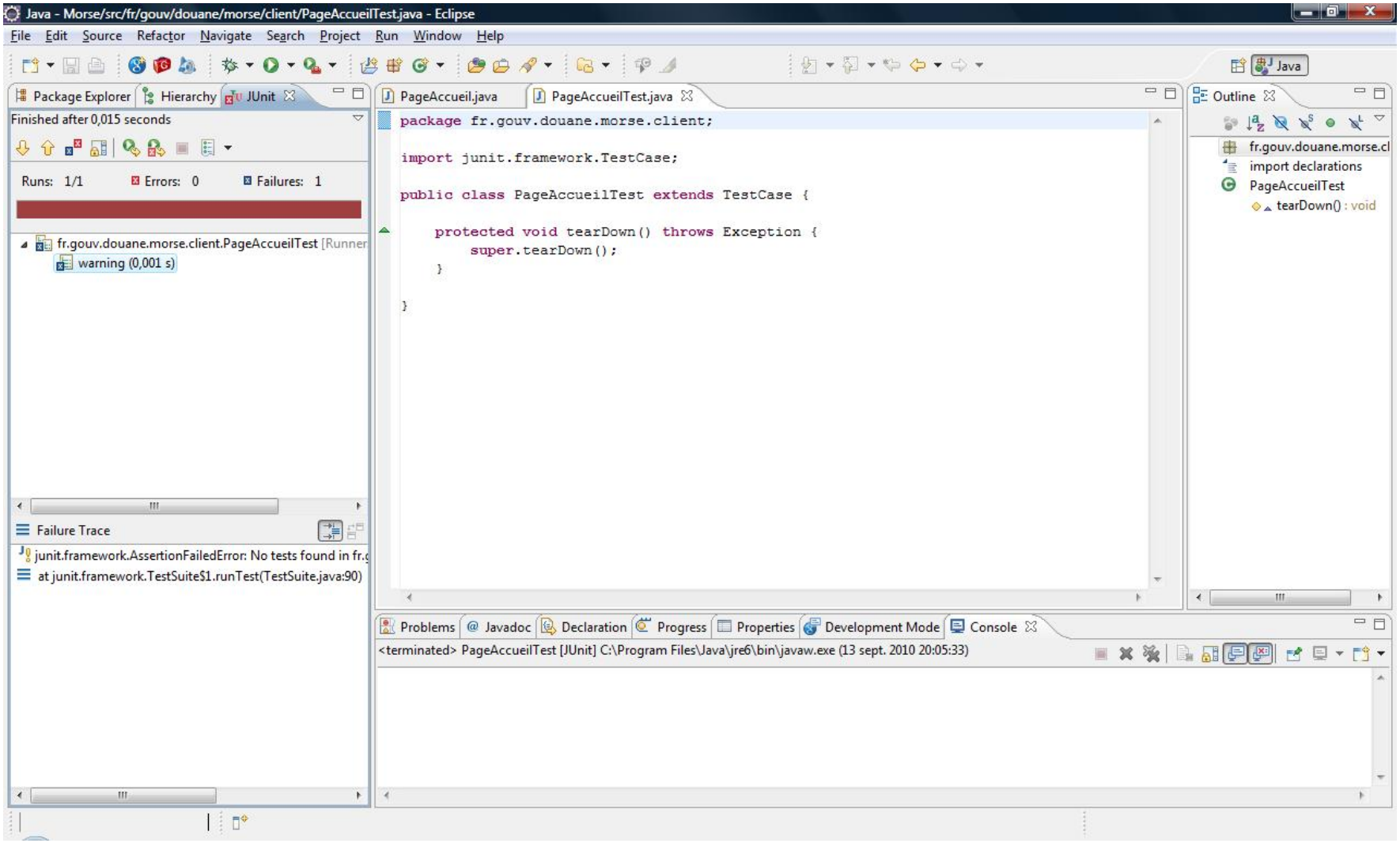
Pour cet exemple j'ai utilisé la classe PageAccueil.



Le TestCase a été créé « PageAccueilTest.java » et on remarque l'adjonction de la bibliothèque Junit3. Le code du TestCase est présenté sur la partie droite de l'écran.



Le test a été exécuté uniquement pour montrer sur l'exemple ce que l'on obtient, il n'y a aucune consistance à ce test.



Junit n'a pas été utilisé dans le cadre du projet MORSE, il est très généralement utilisé pour tester les règles de gestion d'une application. Or, MORSE ne contient pas de règles de gestion dans son fonctionnement si ce n'est pour les états produits. Cependant aucun test unitaire ne peut être effectué sur les états, les tests de cohérences et concordances de ces états étant faits manuellement.

Les autres tests sont ceux effectués par l'utilisateur.
Ces tests permettront avant le recettage de déterminer les bugs non identifiés par le programmeur, bugs générés par une utilisation particulière par exemple, mais aussi pour le recettage afin de déterminer si les besoins de la MOA ont bien été identifiés à travers le développement de l'application.
Cette phase se déroulera début octobre et s'achèvera au recettage de l'application, date à déterminer selon les disponibilités de la la MOA.

Les tests qu'ils soient unitaires ou « utilisateurs » font l'objet de documentations détaillées dans le chapitre suivant.

4.3. Les documents de l'application

4.3.1. Les documents de la phase d'analyse

Les documents de la phase d'analyse ont été décrits dans les chapitres 3.2.1.2 et 3.2.1.3 précédents, ce sont les CDCF et CDCT.

4.3.2. Les documents de la phase de développement

Ils sont constitués par le document de tests unitaires et le document de tests utilisateur.
Leur contenu reprendra les informations détaillées dans le chapitre 4.2 ci-dessus.

4.3.3. Les documents de la phase de recettage

Le document de recettage est élaboré à partir du CDCF, il a pour but de s'assurer que toutes les fonctionnalités demandées par la MOA ont été développées par la MOE.
La validation de ce document vaut recettage de l'application et accord pour sa mise en production.

4.3.4. Les documents de la phase de mise en production

Les documents qui seront repris dans cette phase sont :

- tous les documents techniques de développement de l'application, les sources, les structures de tables, etc ....
- le guide utilisateur
- le document de maintenance de l'application.

Ces documents ne seront finalisés qu'après le recettage et la mise en production de l'application.

4.4. Le bilan de MORSE

L'objectif de la DRGC MMN qui représente la MOA était d'obtenir une application qui lui permette de gérer ses personnels spécialisés en terme de qualification et de réglementations particulières appliquées au domaine maritime et aérien.

Le projet MORSE a été amorcé en septembre 2008 et démarré en fin d'année 2008.
Le cahier des charges fonctionnel a été validé le 8 décembre 2008 dans sa version 3.
Le cahier des charges technique a été validé le 15 octobre 2009 dans sa version 3.
Les maquettes d'écran ont été livrées le 03 novembre 2009 et validées le 01 décembre 2009.
L'application sera livrée fin septembre 2010 pour tests à la DRGC MMN.

Le recettage de l'application par la MOA permettra le cas échéant, sa mise en production par le CID.
À partir de cette étape, les documents relatifs à l'application MORSE seront remis à la MOA et au centre de calcul qui prendra en charge l'application.
Comme évoqué précédemment, l'application est en premier lieu destinée à être utilisée par la DRGC MMN. Si les tests d'utilisation sont concluants, il sera alors envisagé de déployer l'application sur la DRGC MMN, sur la DI de Rouen ou sur tout autre service demandeur.
Cependant, il sera nécessaire d'effectuer les modifications qui ont été décrites dans le chapitre 3.3.6 précédent concernant l'intégration d'Hibernate pour la persistance des objets.

De plus, tous les téléservices repris au sein de la DGDDI et maintenus par l'un des deux centres de calcul (CID ou DNSCE), nécessitent d'intégrer les API qui permettent d'effectuer une authentification par rapport au référentiel RUSH.

5. CONCLUSION

Le projet MORSE est un projet classique d'automatisation d'un processus métier selon un cycle de vie de projet tout aussi classique. La particularité réside dans la conduite à distance du projet, 22 000 km séparant la MOA et la MOE, situation qui contraignait l'organisation du projet, imposant notamment de pallier l'impossibilité de réunir les deux parties pour des comités de pilotage ou de direction ou de former directement les utilisateurs, éléments pourtant essentiels à tout projet.
Ainsi, seules trois réunions à l'initialisation du projet ont eu lieu en fin d'année 2008.
Elles ont permis de poser les prémices du projet et d'interroger la MOA dans le cadre de la définition du CDCF que j'ai élaboré en tant qu'assistance à MOA.
Toutes les validations postérieures entre la MOA et la MOE l'ont été par l'utilisation de la messagerie internet.

Les agents informaticiens de la DGDDI affectés dans les centres de calcul, disposent de tous les outils de développement repris au palier technique et de plateformes de développement, de tests, de validation et de production dont je n'ai pas bénéficié en raison de l'éloignement et des dispositifs de sécurisation des réseaux qui n'ont pas intégré un tel mode de fonctionnement.
Pourtant l'utilisation d'une architecture matérielle et logicielle éprouvée aurait constitué un avantage certain.

Pour le projet MORSE, il m'a fallu apprendre les techniques d'installation et de configuration de la base de données et de l'AGL tout en étant confrontée aux différents problèmes qui peuvent se poser pour une «novice» en ce domaine.
Cet apprentissage peut être considéré comme un avantage par l'acquisition de connaissances techniques désormais dévolues aux services des infrastructures comme dans la plupart des directions informatiques.

L'application des recommandations données par les standards de la sous-direction C pour la conduite de projet ont été respectées au plus proche. Cependant le projet MORSE peut être considéré comme celui d'une petite structure et n'a pas nécessité la mise en œuvre de

moyens ou d'outils de suivi de projet importants tels que des jalons ou un logiciel (MS-Project) par exemple. Le rôle de chef de projet et celui de développeur ayant été tenus par moi même, le pilotage a été réduit à sa plus simple expression.

Dans le cadre du projet MORSE, j'ai préparé toute l'organisation du projet avec les moyens dont je disposais. L'implication de la MOA pour ce projet a certainement contribué à son bon fonctionnement.

Le télétravail[58] sur le projet MORSE n'a été que «partiel» : seule l'utilisation de la messagerie a fait appel aux technologies de l'internet. Cependant la conduite de projet et le développement de l'application MORSE qui a nécessité une organisation spécifique et délocalisée, s'inscrit dans la définition du télétravail de l'Accord National Interprofessionnel (ANI) du 19 juillet 2005 : "*un travail, qui aurait également pu être réalisé dans les locaux de l'employeur, est effectué hors de ces locaux ...*".

Le télétravail a permis sur ce projet « d'effacer » la contrainte de la distance et a été possible grâce à la taille restreinte du projet qui ne nécessitait pas un nombre important de parties prenantes. Sans cela, il aurait été nécessaire de mettre en place une organisation bien plus importante et structurée pour mener ce projet à terme.

Le temps nécessaire consacré à la recherche d'informations pour programmer l'application et l'impossibilité physique de pouvoir acquérir ces connaissances auprès des centres de calcul, ont constitué un inconvénient pénalisant pour le respect des délais initiaux.
L'éloignement géographique ne permet pas les rapprochements physiques facilitant la communication et l'adhésion des parties prenantes du projet. Le management de la communication reste bien trop minimaliste pour un projet de développement d'application.

L'utilisation d'une ressource informatique éloignée des services techniques de la DGDDI est inhabituel et le projet MORSE a permis de démontrer qu'il est possible de réaliser un projet à distance.

Si le télétravail dans le cadre des projets informatiques de la DGDDI venait à se démocratiser, il nécessiterait toutefois la mise en place d'une organisation très particulière

[58] Voir présentation du télétravail en annexe 20

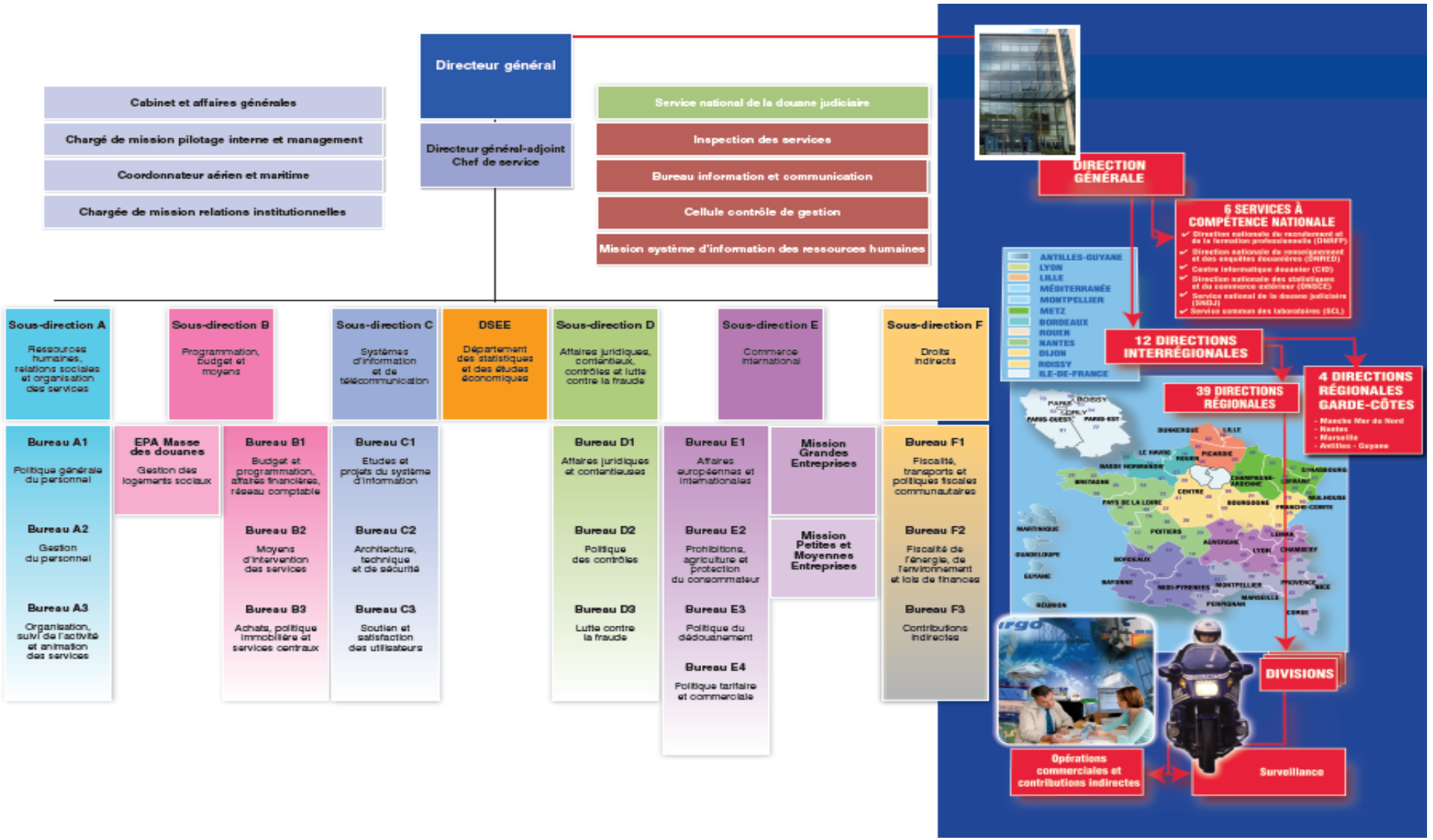
et novatrice en matière de direction et de suivi du projet, au regard des modes de fonctionnements administratifs habituels. La maîtrise des compétences informatiques pourraient être ainsi optimisée en occultant le paramètre géographique.

Le projet MORSE a permis de démontrer qu'il est possible d'effectuer une conduite de projet à distance : de la définition des besoins à la mise en œuvre de l'automatisation avec cependant un retard dû à l'unicité de la ressource.

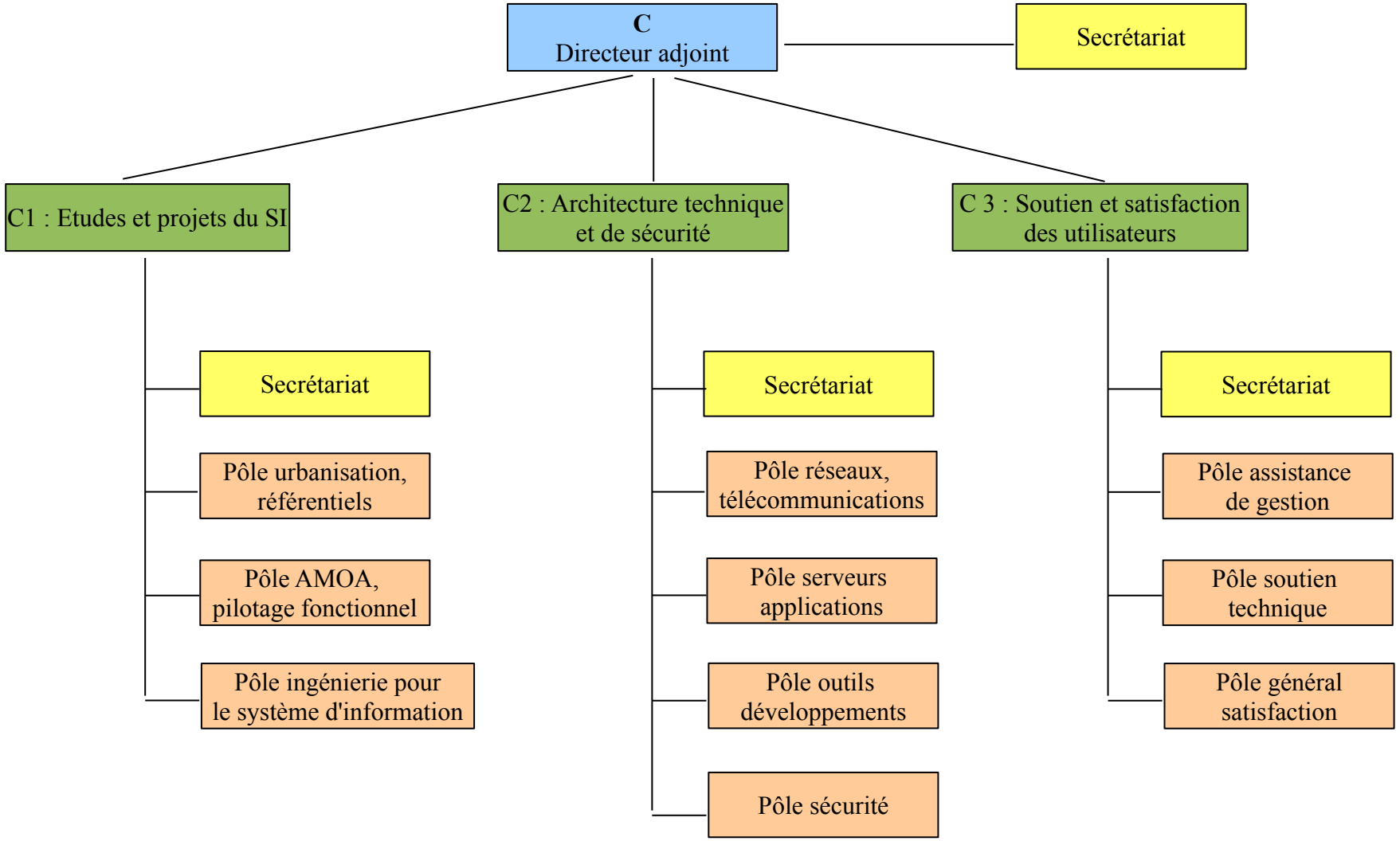
L'aboutissement des travaux et la livraison du téléservice Morse en 2010 tend à prouver que les difficultés inhérentes à une telle situation dans le déroulement d'un projet sont surmontables et qu'il est possible de mener à bien la gestion de projet à distance. C'est une première qui ouvre certainement des pistes d'évolution pour un meilleur emploi des ressources disponibles au sens d'une organisation déterminée.
L'administration ne fait pas usage du télétravail car elle n'a jusqu'à présent pas ressenti un besoin suffisamment important dans ce domaine. Cependant, s'il devait être employé plus systématiquement pour la réalisation de projets qui ne nécessitent pas le rapprochement physiques des équipes, il faudrait alors envisager la définition d'une méthodologie et la mise en place de moyens nécessaires à un emploi optimal des ressources disponibles.

ANNEXES

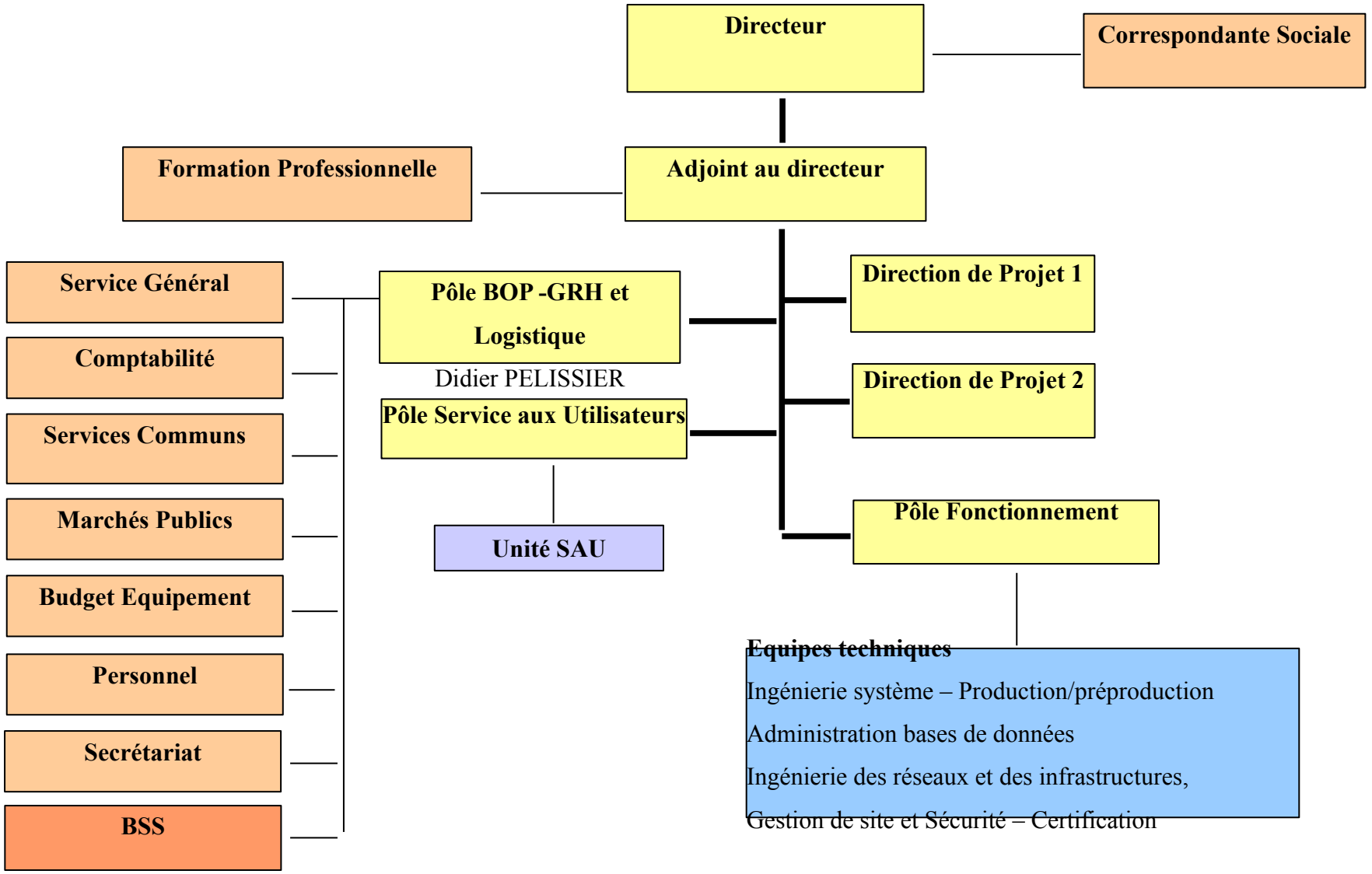
Annexe 1 : Organigramme de la DGDDI



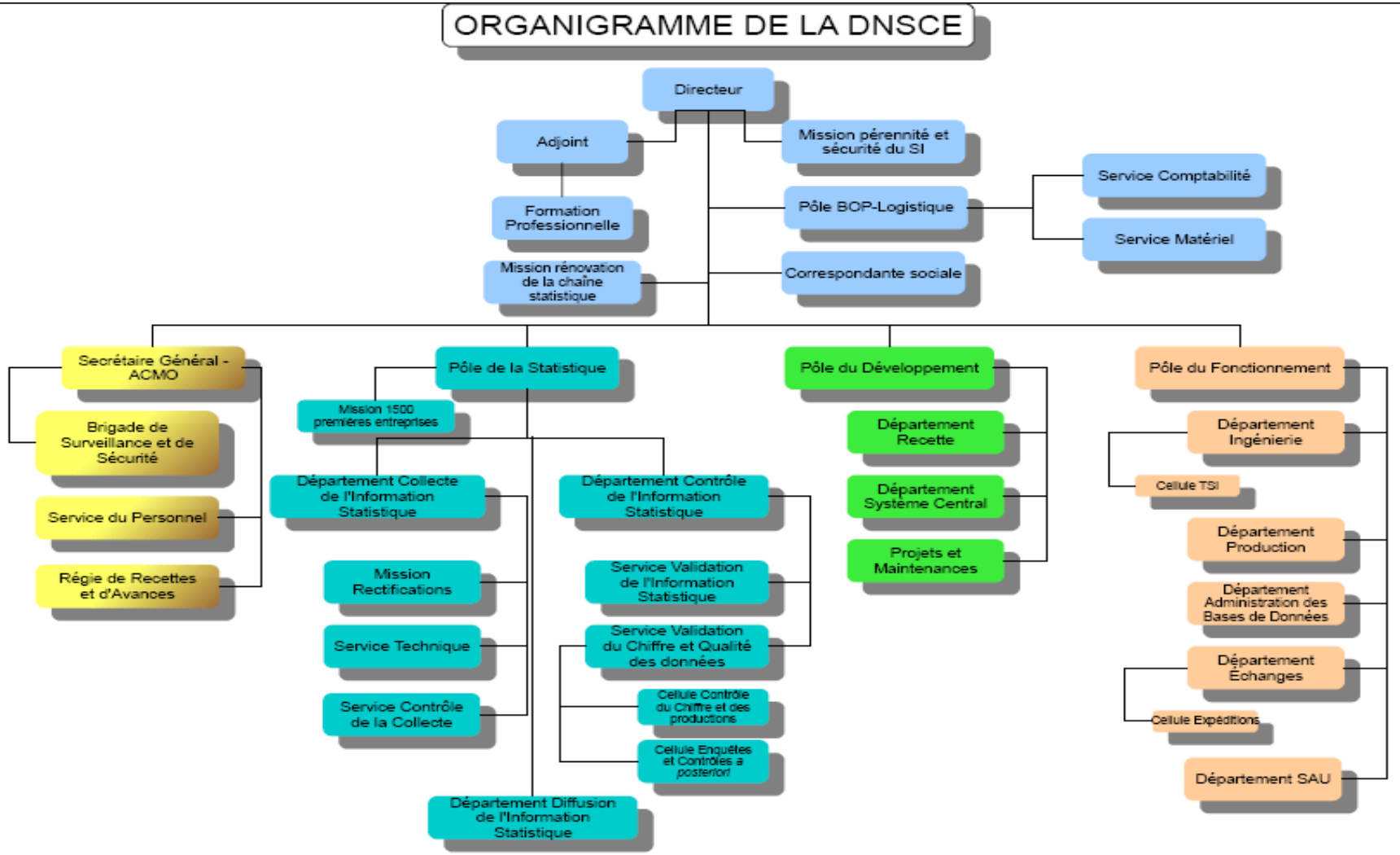
Annexe 2 : **Organigramme de la sous-direction C**



Annexe 3 : **Organigramme du CID**



Annexe 4 : **Organigramme de la DNSCE**



Annexe 5 : Le cadre réglementaire des qualifications informatiques au sein du Ministère

L'informatique dans l'administration voit le jour dès les années 1960 mais c'est à partir des années 1970 que l'essor de l'informatique rend nécessaire l'établissement d'un régime particulier adapté aux impératifs de l'évolution incessante.
Ainsi, plusieurs textes ont été votés pour fournir un cadre réglementaire aux personnels informaticiens de la fonction publique :

- La loi n° 70-1211 du 23 décembre 1970 relative à la situation des fonctionnaires affectés au traitement de l'information.
- Le décret n° 71-342 du 29 avril 1971 relatif à la situation des fonctionnaires affectés au traitement de l'information.
- Le décret n° 71-343 du 29 avril 1971 relatif aux fonctions et au régime indemnitaire des fonctionnaires de l'Etat et des établissements publics affectés au traitement de l'information.
- L'arrêté du 10 juin 1982 relatif aux programmes et nature des épreuves des concours et examens portant sur le traitement automatisé de l'information.
- La loi n° 83-26 du 19 janvier 1983 modifiant l'ordonnance 59244 du 04 février 1959 relative au statut général des fonctionnaires.

Mais l'évolution constante des technologies informatiques et l'avènement de l'internet ont obligé la fonction publique à reconsidérer les « métiers » informatiques existants pour s'adapter plus précisément à l'émergence des nouveaux métiers informatiques tels que technicien réseau, administrateur de base de donnée, ingénieur sécurité, cogniticien (réalise les systèmes experts), ergonome ou encore infographiste.

Suite à un groupe de travail et à la reconnaissance du besoin de création de nouveaux métiers, l'Assistant Utilisateur sera rattaché à la qualification d'Analyste (AAU) ainsi qu'à celle de Pupitreur (PAU), l'Expert Réseau et le Concepteur Réalisateur d'Application seront rattachés à la qualification de Programmeur Système d'Exploitation (PSE-ER et PSE-CRA).
Ces rattachements ont permis de rester dans le cadre du décret n° 71-343 du 29 avril 1971 comme demandé par le Ministre, en créant de nouveaux métiers sans avoir à créer de nouvelles qualifications. Seules les épreuves d'accès à ces qualifications ont été aménagées afin de répondre aux spécificités des nouveaux métiers.

Annexe 6 : Les métiers informatiques et qualifications associées

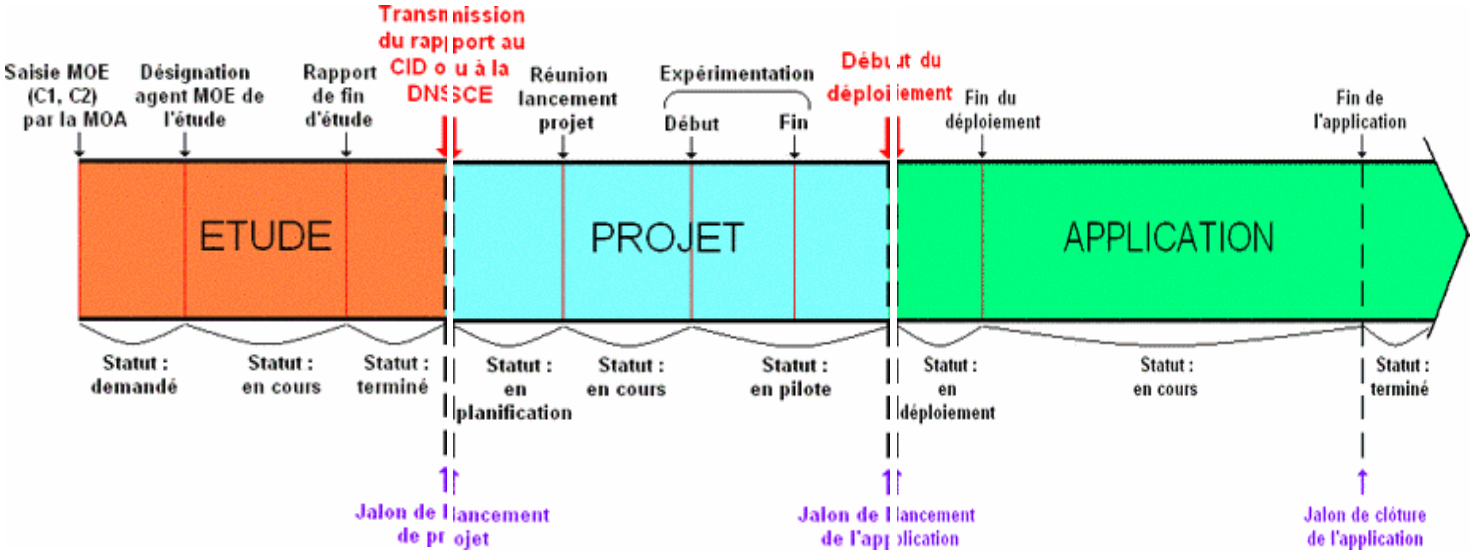
Les concours et examens professionnels pour l'obtention d'une qualification informatique ne sont ouverts qu'aux catégories administratives correspondantes. Ainsi pour avoir la qualification d'analyste, il est nécessaire d'être de catégorie A (voir tableau ci-dessous).

CATEGORIE	QUALIFICATION	REMARQUES
A	Chef de projet	
	Analyste	
	Analyste Assistant Utilisateur (AAU)	Nouvelle qualification
B	Chef programmeur	
	Programmeur	
A ou B	Chef d'exploitation	
	Programmeur Système d'Exploitation (PSE)	
	Programmeur Système d'Exploitation – Expert Réseau (PSE-ER)	Nouvelle qualification
	Programmeur Système d'Exploitation – Concepteur Réalisateur d'Application (PSE-CRA)	Nouvelle qualification
C	Moniteur de dactylocodage	Qualification abandonnée
	Dactylocodeur	Qualification abandonnée
	Agent de traitement	Qualification abandonnée
	Bibliothécaire de support (médiathécaire)	
	Agent de façonnage	
B ou C	Pupitreur	
	Pupitreur Assistant Utilisateur (PAU)	Nouvelle qualification
	Préparateur de travaux	Qualification abandonnée

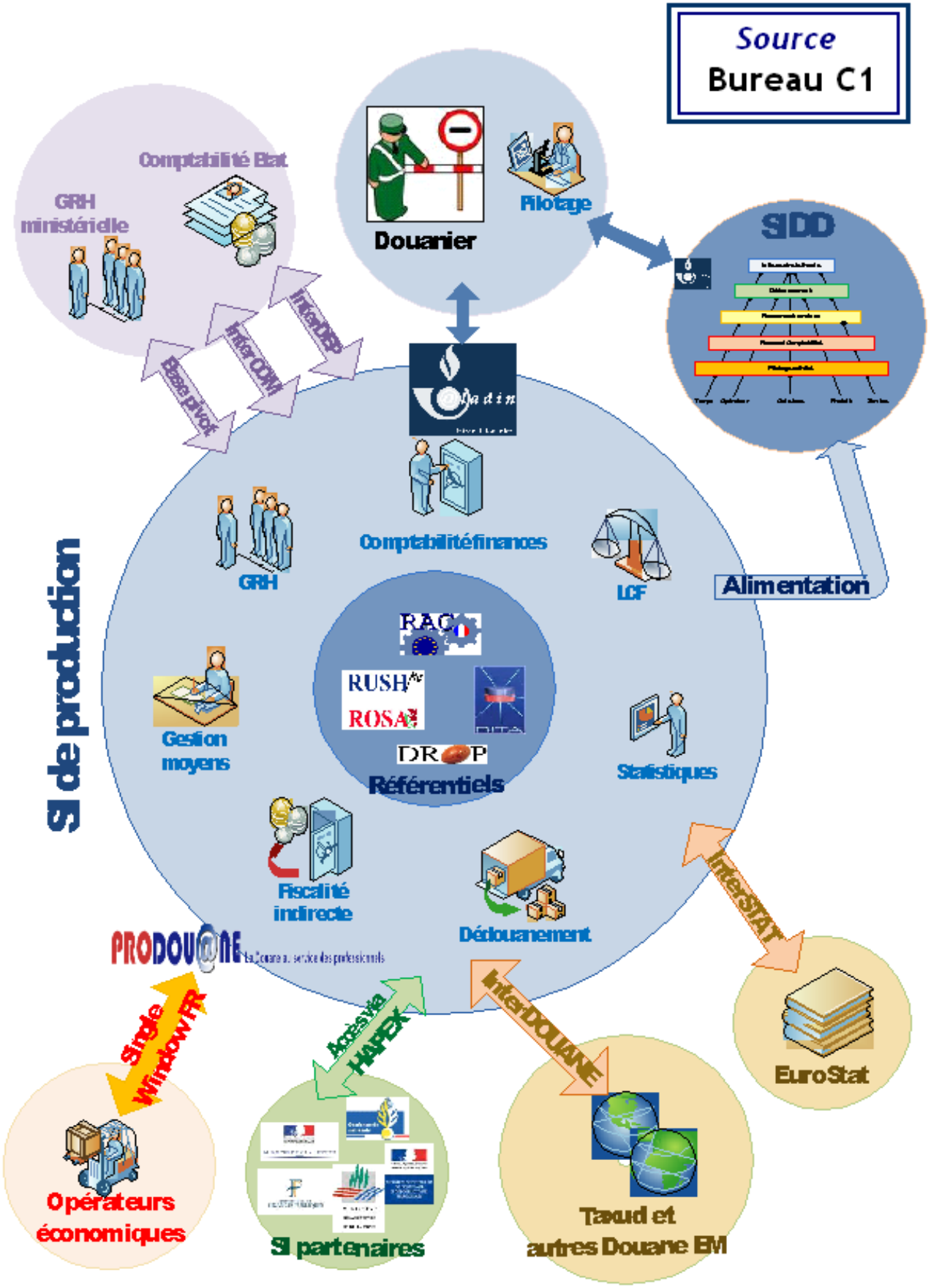
Les différentes catégories, au nombre de trois, qui existent au sein de l'administration d'Etat sont :

- ●Catégorie A : obtenue par un concours exigeant un niveau d'études supérieures d'au moins trois années en externe, cette catégorie est hiérarchiquement supérieure aux deux autres et correspond à des fonctions de conception, de direction et d'encadrement.
- ●Catégorie B : en concours externe, le niveau d'études exigé est baccalauréat. Les fonctions dévolues à cette catégorie sont des fonctions d'application et de rédaction.
- ●Catégorie C : un Diplôme National du Brevet est nécessaire à l'obtention de cette catégorie par concours externe. Elle est destinée à des fonctions d'exécution.

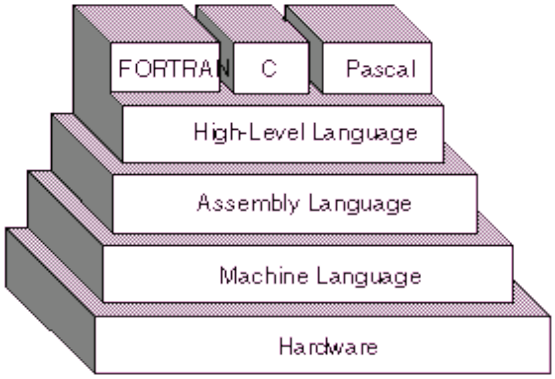
Annexe 7 : Le cycle de vie d'un projet informatique à la DGDDI



Annexe 8 : **Le système d'information de la DGDDI**



Annexe 9 : Les générations de langages informatiques.



•**Première génération** : langage machine qui constitué uniquement de nombres compréhensible par la machine

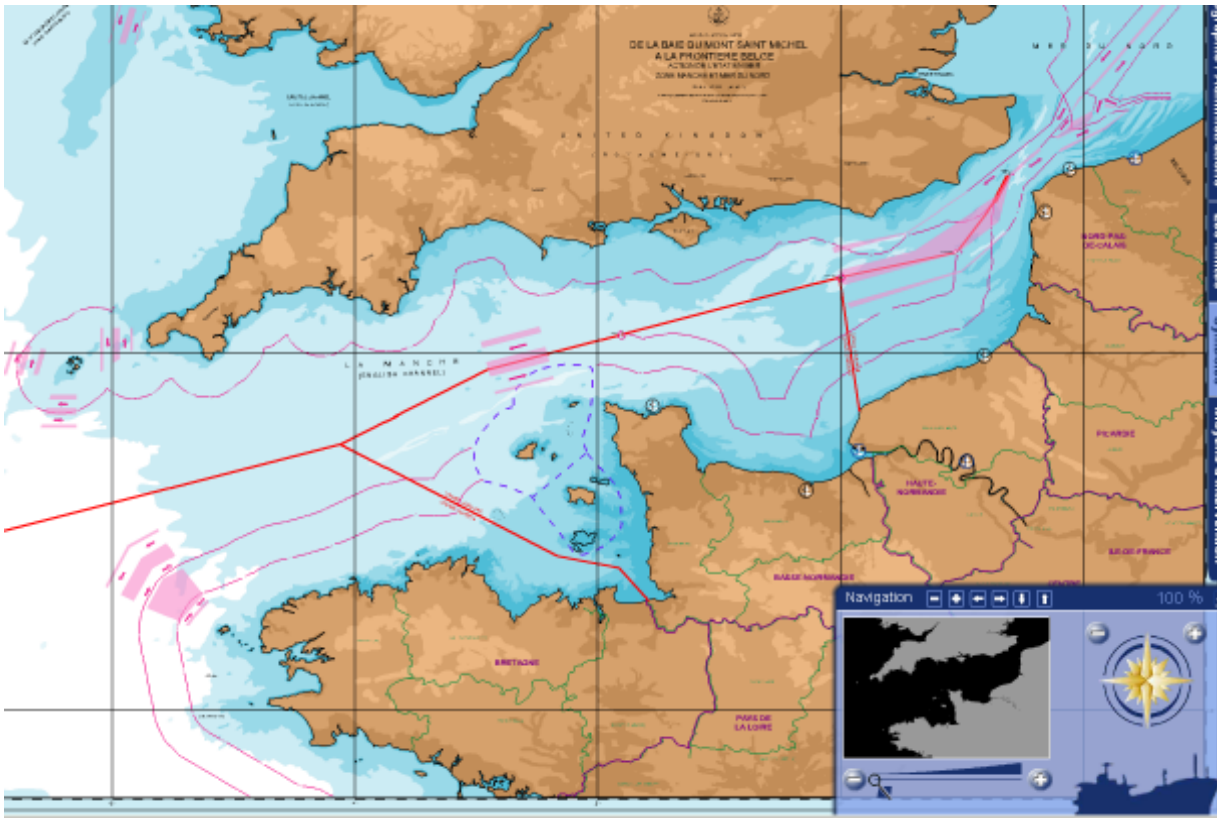
•**Deuxième génération** : langage permettant au programmeur d'utiliser des noms plutôt que des nombres (Assembleur)

•**Troisième génération :** langages de programmation de haut niveau tel que COBOL, C++, JAVA

•**Quatrième génération :** langages de programmation proches du langage humain, souvent utilisés pour les accès aux bases de données (SQL)

•**Cinquième génération :** langages utilisés pour l'intelligence artificielle pour simuler le comportement humain par un ordinateur (LISP, PROLOG) ou les réseaux neuronaux pour tenter de reproduire le fonctionnement d'un cerveau humain (reconnaissance vocale)

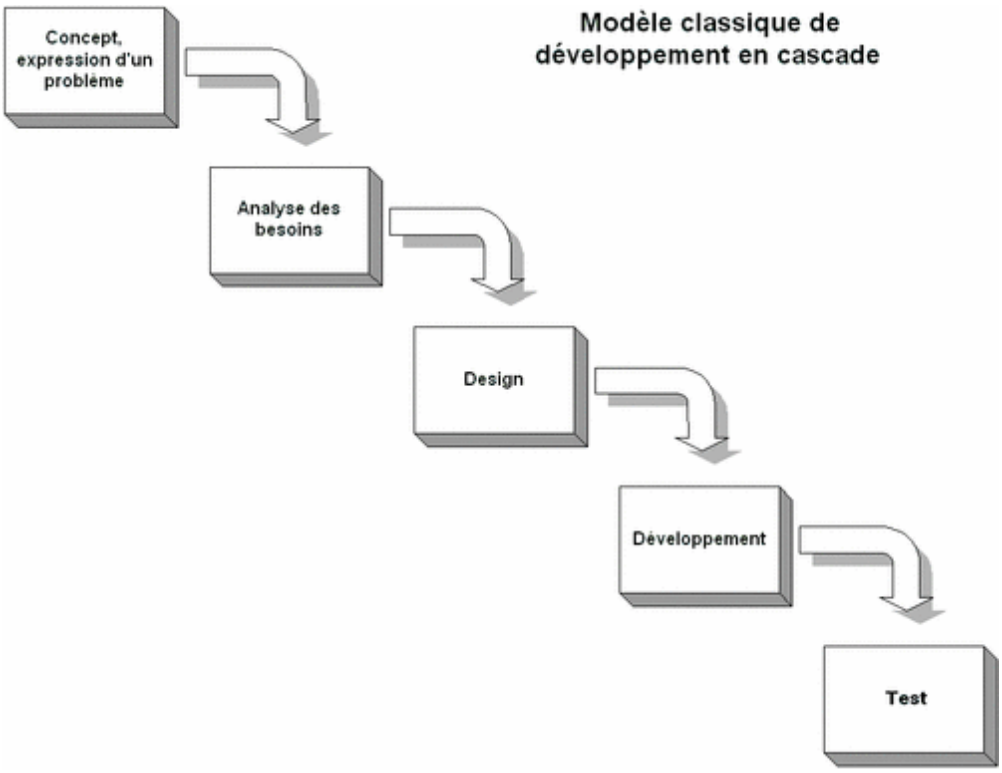
Annexe 10 : Zone d'action de la DRGC-MMN



Zone de compétence MMN: limites des eaux territoriales et zone contigüe (traits rose), Iles Anglo-normandes (pointillés bleu), limites de zones des CROSS Gris-Nez et Jobourg (traits rouge) (source Préfecture Maritime MMN)

Annexe 11 : Les différents modèles de cycle de vie

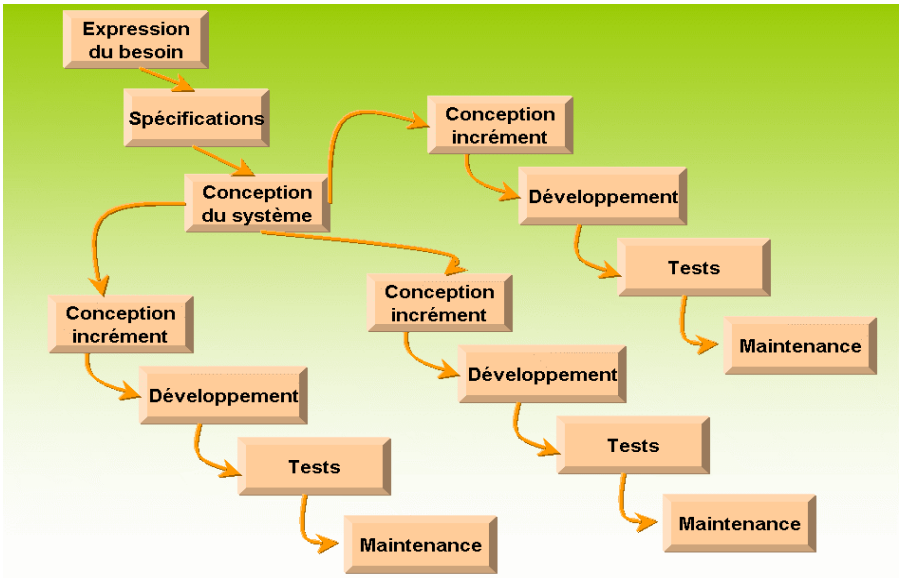
Ainsi, on définit **le modèle en cascade** issu du BTP[59], qui part du principe qu'une action doit être finie avant de pouvoir entamer la prochaine (ex : construction des fondations avant les murs). Ce modèle oblige à revenir au début si une anomalie était constatée.



Le modèle incrémental ou loti est utilisé pour gérer les projets de développement de grands systèmes. Il découpe le système en domaines traités individuellement sur le modèle en cascade. Une des idées du modèle en incrémental est de gérer la complexité.

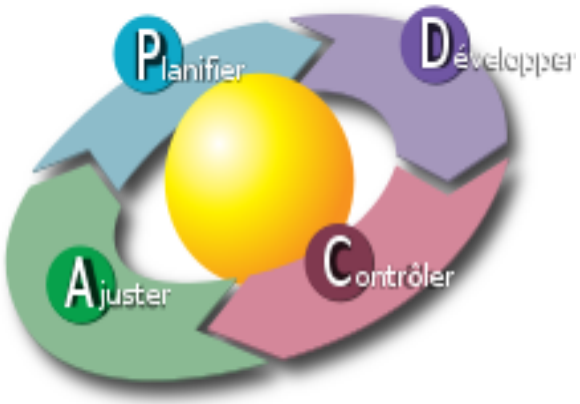
C'est un modèle particulièrement adapté aux grands projets. Les domaines doivent être suffisamment découplés sinon certains incréments devront attendre que les incréments avec lesquels ils sont liés, soient suffisamment développés. Il est du ressort des MOA de vérifier le couplage des domaines.

[59] Bâtiment et Travaux Publics

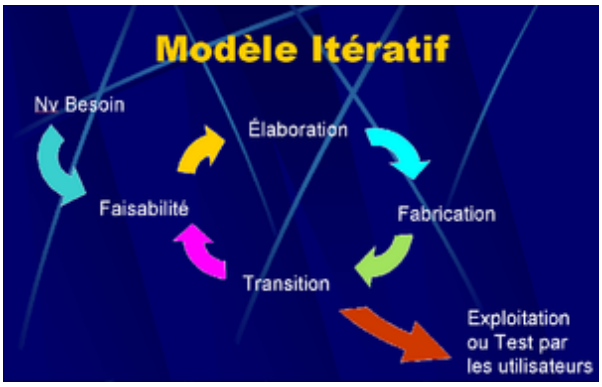


Le **modèle itératif** permet de séparer les activités des produits. On applique un cycle de type roue de Deming pour la production d'une documentation, d'un code, d'un test, ...

La roue de Deming permet d'atteindre ou d'améliorer une qualité de production en suivant un cycle «Plan-Do-Check-Act (PDCA)».

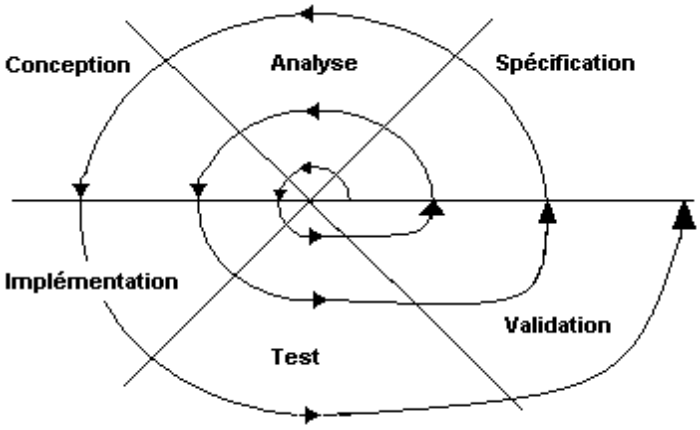


L'objectif est de livrer au plus tôt quelque chose qui sera testé par le client. Ainsi plusieurs itérations peuvent être faites sur une documentation.



Le modèle en spirale a été défini par Barry Boehm en 1988 dans son article "A Spiral Model of Software Development and Enhancement"
L'emphase de ce modèle et mise sur la réduction des risques. Il est souvent utilisé pour les gros projets complexes.

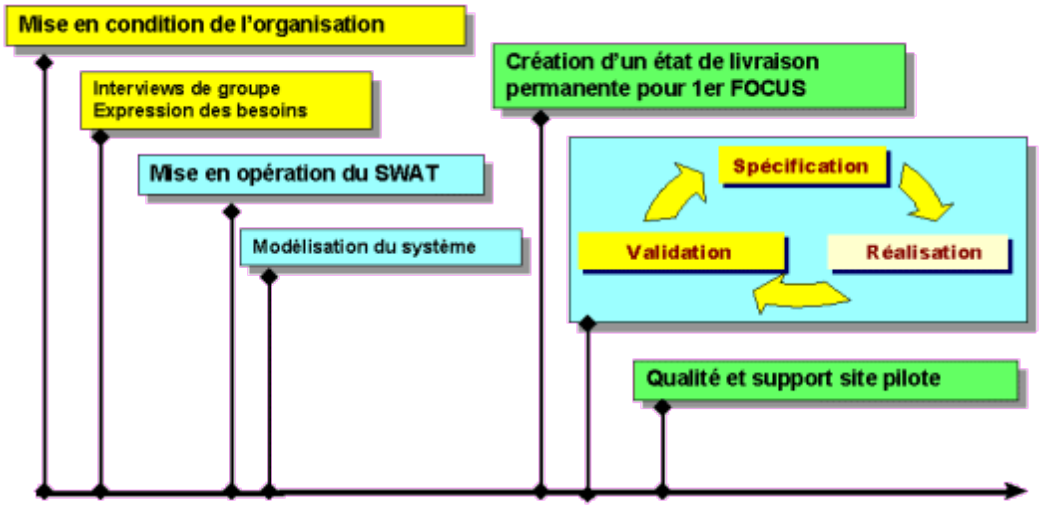
Il reprend les différentes étapes du cycle en V en implémentant les versions successives. Le cycle est déroulé plusieurs fois et propose un produit de plus en plus complet.



La méthode de développement rapide d'applications (**modèle RAD** - Rapid Application Development), est la première méthode de développement de logiciels où le cycle de développement est en rupture fondamentale par rapport à celui des méthodes antérieures dites « en cascade ».
Le cycle de vie RAD est employé lorsque l'implication forte de l'utilisateur est nécessaire. Une condition sine qua non pour mettre en œuvre un cycle RAD est que l'équipe projet sache parfaitement maîtriser l'AGL employé pour garantir un passage rapide du concept à la mise en œuvre.

Le cycle RAD comporte 3 phases :

- Cadrage (Joint Requirement Planning) qui couvre l'analyse des besoins, le périmètre et la planification de l'itération
- Conception (Joint Application Design) qui couvre la conception, description et organisation des données et des traitements avec les utilisateurs
- Construction (Construction Assistance Team) qui couvrent le développement et les tests.



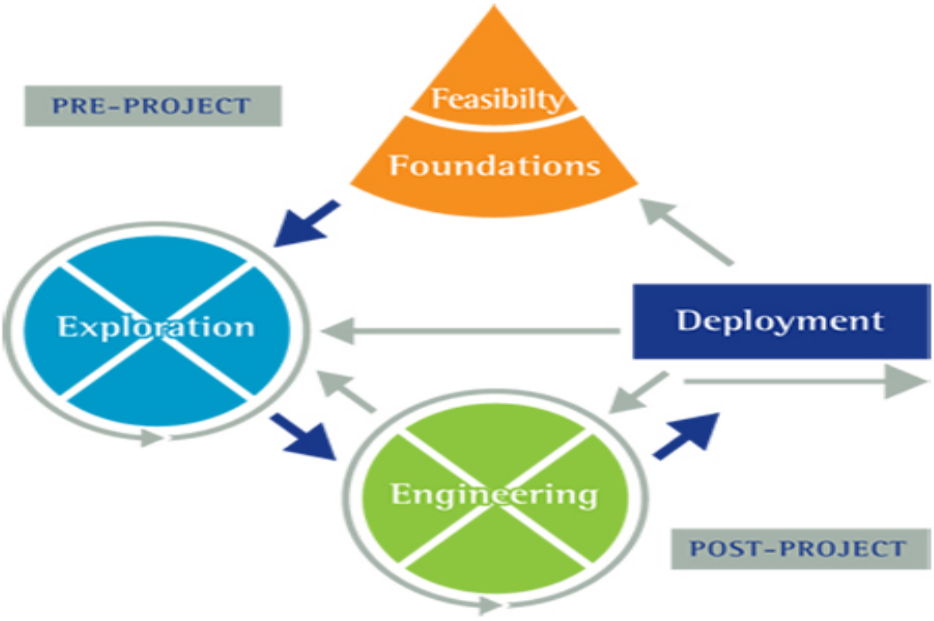
La méthode du **prototypage** permet de construire un produit (logiciel) pour tester son utilisation et son utilisabilité auprès des utilisateurs. Elle permet de simuler des fonctions avec des données fictives ou réelle.

Le prototype se rapproche au plus près du produit final et peut être ré-itérer plusieurs fois, cepedant la méthode est coûteuse puisqu'elle nécessite un environnement identique à celui de production (serveur, poste de travail, logiciels, ...)

La méthode DSDM (Dynamic Systems Development Method) est une évolution de la méthode RAD et a été développée dans le milieu des années 90. Elle fait partie de la catégorie des méthodes agiles (méthode basée sur un développement itératif et incrémental où les besoins et les solutions évoluent en collaboration étroite avec les membres de l'équipe projet).

La méthode DSDM s'appuie sur 9 principes de base :

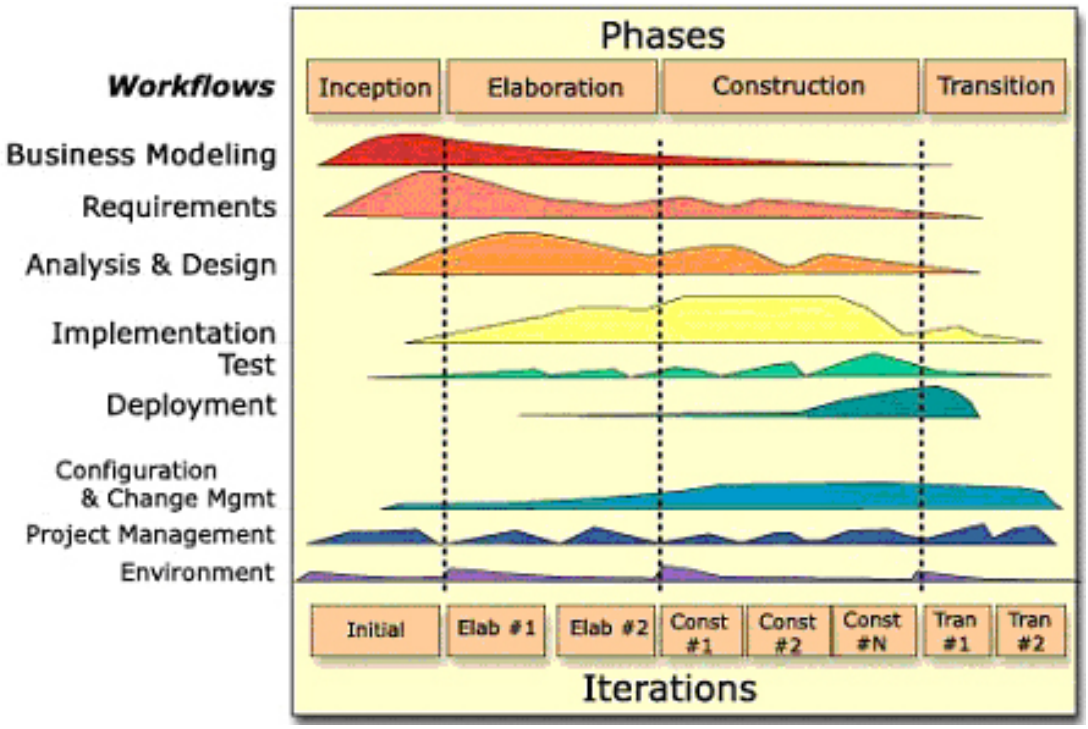
- Implication des utilisateurs durant tout le cycle de développement, ils font partie intégrante de l'équipe projet.
- Autonomie : l'équipe projet prend les décisions concernant l'évolution des besoins.
- Visibilité du résultat : l'application est livrée le plus souvent possible afin de permettre un feed-back rapide.
- Adéquation : l'objectif est de livrer une application en adéquation avec le besoin metier du client.
- Développement itératif et incrémental.
- Réversibilité : toute modification effectuée durant le développement doit être réversible.
- Synthèse : un schéma directeur défini de manière préalable, fixe les grandes lignes du projet, notamment son périmètre.
- Tests : en continu durant tout le développement.
- Coopération : les acteurs du projet doivent faire preuves de souplesse concernant les modifications des fonctionnalités demandées.



La méthode RUP (Rational Unified Process) est la plus connue de ces types de processus. Il est proche des méthodes agiles, se concentre sur ce qui est d'un intérêt direct pour le client et observe un processus itératif guidé par les risques.

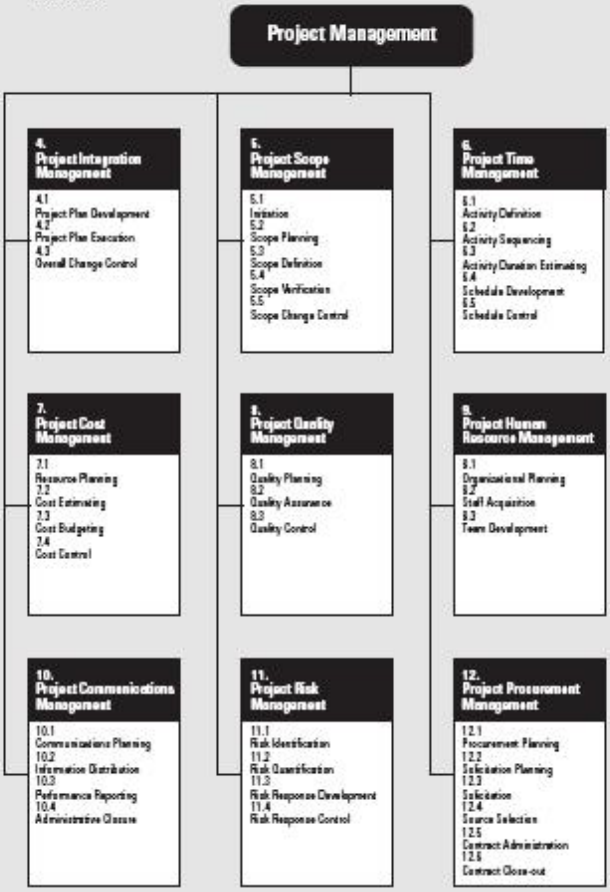
La méthode RUP se divise en 4 phases :

- *L'inception* est une phase de vérification du projet. On aura défini les tâches à accomplir, les outils à utiliser et les risques possibles qui seront éliminés au plus tôt.
- *L'élaboration* cette seconde phase permet d'affiner la phase précédente. C'est une phase où une première version (ou l'élaboration d'un prototype) va permettre d'évaluer les risques techniques afin de les réduire.
- *La construction*. C'est la phase d'implémentation proprement dite. Les fonctionnalités du projet seront développées en utilisant un certains nombre d'itérations fonctionnelles. Le logiciel livré au fur et à mesure sera de plus en plus enrichi.
- *La transition.* On prépare la phase de déploiement (la documentation, la formation des utilisateurs, la préparation des outils marketings,...). Cette phase permet également de préparer les prochaines évolutions liées à la maintenance du logiciel. La phase de transition dure donc toute la vie du logiciel une fois celui-ci réalisé.


Phases
Workflows
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration
& Change Mgmt
Project Management
Environment
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #N
Tran #1
Tran #2
Iterations

Annexe 12 : Domaines du PMBOK

Figure 1–1. Overview of Project Management Knowledge Areas and Project Management Processes



Annexe 13 : Plan type du cahier des charges fonctionnel selon la norme AFNOR NF X50-151

1. Présentation générale du problème

1.1 Projet

1.1.1 Finalités

1.1.2 Espérance de retour sur investissement

1.2 Contexte

1.2.1 Situation du projet par rapport aux autres projets d e l'entreprise

1.2.2 Etudes déjà effectuées

1.2.3 Etudes menées sur des sujets voisins

1.2.4 Suites prévues

1.2.5 Nature des prestations demandées

1.2.6 Parties concernées par le déroulement du projet et ses résultats (demandeurs, utilisateurs)

1.2.7 Caractère confidentiel si il y a lieu

1.3 Enoncé du besoin (finalités du produit pour le futur utilisateur tel que prévu par le demandeur)

1.4 Environnement du produit recherché

1.4.1 Listes exhaustives des éléments (personnes, équipements, matières…) et contraintes (environnement)

1.4.2 Caractéristiques pour chaque élément du l 'environnement

2. Expression fonctionnelle du besoin

2.1 Fonctions de service et de contrainte

2.1.1 Fonctions de service principales (qui sont la raison d 'être du produit)

2.1.2 Fonctions de service complémentaires (qui améliorent, facilitent ou complètent le service rendu)

2.1.3 Contraintes (limitations à la liberté du concepteur-réalisateur)

2.2 Critères d ’appréciation (en soulignant ceux qui sont déterminants pour l ’évaluation des réponses)

2.3 Niveaux des critères d ’apprécpiation et ce qui les caractérise

2.3.1 Niveaux dont l ’obtention est imposée

2.3.2 Niveaux souhaités mais révisables

3. Cadre de réponse

3.1 Pour chaque fonction

3.1.1 Solution proposée

3.1.2 Niveau atteint pour chaque critère d ’appréciation de cette fonction et modalités de contrôle

3.1.3 Part du prix attribué à chaque fonction

3.2 Pour l ’ensemble du produit

3.2.1 Prix de la réalisation de la version de base

3.2.2 Options et variantes proposées non retenues au cahier des charges

3.2.3 Mesures prises pour respecter les contraintes et leurs conséquences économiques

3.2.4 Outils d ’installation, de maintenance … à prévoir

3.2.5 Décomposition en modules, sous-ensembles

3.2.6 Prévisions de fiabilité

3.2.7 Perspectives d ’évolution technologique

Annexe 14 : Cahier des Charges Fonctionnel du projet MORSE

Annexe 15 : Cahier des Charges Technique du projet MORSE

Annexe 16 : Palier technique de la DGDDI au 11/12/07

SYSTEME D'INFORMATION ET DE TELECOMMUNICATION DOUANE

DG-C2 – 11/12/2007 — **PALIER TECHNIQUE TOUT INTERNET – version 2007B** — http://dg-c2.douane — 1

POSTE DE TRAVAIL

	Ancien palier	Palier actuel
Système d'exploitation		XP Pro SP2 (2/2006)
Environnement JAVA *	J2RE 1.4.2 (6/2005)	J2RE 1.5.0_06 (2/2006)
Navigateur Internet	IE 6 SP1 (6/2003)	IE 6 SP2 (2/2006)
Outil client de messagerie	OE 6 SP1 (6/2003)	OE 6 SP2 (2/2006)
Bureautique **	OOO 1.1.1 (1/2004)	OOO 2.0.2 (2/2006) + extensions douane
Format bureautique	formats OOO V1	OpenDocument (11/07)
Création PDF		PDF Creator 0.93
Visualisation	Flash Player 9.0.16.0	Flash Player 9.0.28.0 (2/2007)
	Acrobat Reader 7.0.5 (6/2005)	Acrobat Reader 7.0.8 (2/2007)
		Visualisateurs Office (11/7) V2003
Compression	PowerArchiver 6/2000	Win XP (par défaut) 7ZIP4.42 (en secours)
Prise de main à distance		Bureau à distance ou VNC 4.12 (2/2004)
Transferts volumineux		Grokoli 1.07 (8/2004) future version / VS8.5

SECURITE POSTE CLIENT

	Ancien palier	Palier actuel
Antivirus poste client	VirusScan 7.1	Viruscan 8.5 (1/2008)
Carte à puce		Carte CADO (2/2007) lecteur pcmcia/usb

RESEAUX LOCAUX ET ACCES POSTES DE TRAVAIL

	Ancien palier	Palier actuel
Câblage		Catégorie 5 (6/2003) 100 Mb/s
WIFI	IEEE 802.11b (6/2003) IEEE 802.11g (2/2006)	Wifi fortement déconseillé sauf usages particuliers (11/7)
CPL	Homeplug 1.0 (12/2006) minimum 85 Mb/s	CPL abandonné sauf usages particuliers (11/7)
Réseau mobile	carte EDGE (2/2006)	via PDA (2/2007)
Analyse LAN **	Ethereal 0.10 (2/2006)	wireshark 0.99 (2/2007)
**		NMAP 3.7 (2/2006)

COMPATIBILITE APPLICATIONS - EXPLORATEURS

	Ancien palier	Palier actuel
NIVEAU PAGES WEB		HTML 4.01 (XHTML) (6/2003)
SCRIPT sur Pages WEB		Javascript 1.5 ECMA-262 r3 (6/2003)
ACCES ALADIN (intranet)		XP SP2 + IE 6 + Firefox (2/2007)
ACCES PRODOUANE (internet)		IE 5 + Firefox 1.0

MASTERISATION

Ce palier de spécifications s'applique à tous les nouveaux systèmes à développer, à intégrer dans l'infrastructure de la Douane, et représente la cible de portage des applications existantes. Ancien palier désigne les versions antérieures encore installées. * indique logiciel libre, ** : support spécifique par le marché LL *Premier palier: 6/2003*

les PC sont acquis avec la prestation de mastérisation : duplication du master douane sur les postes en usine avant livraison et fourniture d'un support (DVD) de restauration du poste mastérisé au moment de la 1ere livraison (au moins 2 supports par direction).

CONFIGURATION MATERIELLE ORDINATEUR DE BUREAU

Micro-ordinateur compatible PC, Boitier format miniTour

Processeur de type X86 en 32 bits.

RAM 1 GO, extensible à 2 GO avec emplacement libre

Un disque dur de capacité de 80 GO

Lecteur/graveur de DVD ROM

un lecteur disquette 3,5 pouces ;

un clavier 105 touches AZERTY français disposant du symbole de l'euro ;

une souris à molette ou optique;

un contrôleur Ethernet 100 Mbits/s intégré sur la carte mère ou carte Ethernet 100 Mbits/s PCI ;

un contrôleur audio ou carte son intégrée ;

un contrôleur vidéo intégré ;

un port souris, port clavier (mini-din) et port écran ;

Ports 1 parallèle, 1 série et 4 USB2.0, dont 2 sur la face avant

1 connecteur d'extension PCI libre ;

En option: lecteur interne de cartes à puce

Moniteur plat LCD 19 pouces (résolution native = 1280x1024).
NB : les applications intranet sont conçues et optimisées pour une fenêtre d'affichage de 1024*768.

Documentation technique en français

CONFIGURATION MATERIELLE ORDINATEUR PORTABLE

Processeur de type X86 en 32 bits. Doit permettre à la configuration d'obtenir un résultat minimum au test demandé : actuellement 100 au test Sysmark 2004

RAM 1 GO sur une seule barrette standard, extensible à 2 Go. 1 emplacement mémoire disponible.

Disque dur 80 GO

Ecran TFT 13 pouces affichant au minimum une résolution de 1024x768; peut être 16/10

Poids maximum : 2,3 kg pour une autonomie de 3 heures minimum en utilisation bureautique porté à 2,7 kg pour une autonomie de 6 heures.

Circuit son intégré

Clavier AZERTY français avec sérigraphie "€"

Dispositif de pointage intégré

graveur de DVD

Circuit réseau ETHERNET 10/100, conforme aux normes en vigueur

un port PCMCIA (PC CARD) de type II ou de type III

Ports 2 x USB 2.0, //, série, clavier, souris, écran externe

Option : batterie supplémentaire, kit allume-cigare

Sacoche de transport

Documentation technique en français

Kit de reconstitution de l'environnement initial.

CONFIGURATION MATERIELLE ORDINATEUR CRISTAL MOBILE (Ancien Palier)

Par rapport à l'ordinateur portable, possibilité de disposer d'une carte réseau mobile et d'un lecteur de carte à puce simultanément:

- soit 2 emplacements PCMCIA type 1;
- soit 1 PCMCIA type 1 pour carte EDGE + Lecteur de cartes intégrés;
- soit 1 PCMCIA type 1 pour lecteur de carte + carte EDGE intégrée.

TERMINAUX MOBILES (2007)
PDA EDGE NOKIA N61

COMPLEMENTS-GARANTIE
Les micros et imprimantes disposent d'une garantie sur site de 3 ans.

SERVEURS DE MESSAGERIE ET D'ANNUAIRE

	Ancien palier	Palier actuel
SERVEURS DE MESSAGERIE Interpesonnels		Sun IPlanet V5.1 SP2 messaging server (2/2004)
Filtrage messagerie		Trend IMSS5.7 (2/2007) Trend TMCM 3.0 (2/2007)
SERVEURS D'ANNUAIRE Interpersonnels		Sun Iplanet V5.2 directory server (2/2004)
Serveurs de messagerie* *		Postfix 2.0 (2/2006)
Serveur POP/IMAP **		CYRUS IMAP 2.2 (2/2006)
Annuaires **		OpenLDAP 2.2 (2/2006)

SERVEURS APPLICATIFS

	Ancien palier	Palier actuel
OS		AIX 5.2 (6/2003)
**	LINUX RED HAT AS3 (2/2004)	LINUX RED HAT AS4 (32 bits) (2/2006)
		LINUX RED HAT 64 bits pour Oracle /LARI (11/7)
	Windows 2000 SP3 (2/2004)	Windows 2003 SPx (11/2007)
SHELL **		BASH 2.0.5 (tous OS) (2/2006)
SGBDR	Oracle 9i (6/2003)	Oracle 10g (2/2006)
	SQL Server 2000 SP3 (6/2003)	SQL Server 2005 SPx (11/2007)
Infocentres		Oracle Discoverer 2007 (11/2007
SERVEURS WEB **	Apache 1.3 (6/2003)	Apache 2 (2/2006) mod_jk mod_ssl ATWStats 6.4 (2/2007)
**		OpenSSL 0.9.7 (2/2007)
		IIS 5/Windows (6/2003)
WEB/Servlet **	TOMCAT 4.1 (6/2005)	TOMCAT 5.5 11/7)
J2EE **	Weblogic 8.1 SP2 JBOSS 3.2.5 (2/2006)	JBOSS 4.0.4 (11/7) XCommonJ 0.1.2 (patchs Weblogic) (11/2007

AUTRES FONCTIONS SERVEURS

	Ancien palier	Palier actuel
Serveur d'émulation GCOS-IBM		ILEX (6/2003)
Infocentres	Business Objects WEBI 6.0 (6/2003)	Outils ORACLE (2/2007)
Serveur de temps **		Ntp 4.2 (2/2006)
Transfert de fichiers SMB* *		Samba 3.0 (2/2006)
Synchronisation **		Rsync 2.6.5 (2/2006)
Partage de Fichiers UNIX **		NFS 1.0.5 (2/2006)
Remontée et stockage d'événements **		Rrdtool 1.2.10 (2/2006)
Statistiques systèmes **		sysstat 6.0.1 (2/2006)
Récupération de pages WEB *		Wget 1.8.2 (2/2006)

Ce palier de spécifications s'applique à tous les nouveaux systèmes à développer, à intégrer dans l'infrastructure de la Douane, et représente la cible de portage des applications existantes. Ancien palier désigne les versions antérieures encore installées. * indique logiciel libre, ** : support spécifique par le marché LL *Premier palier: 6/2003*

ADMINISTRATION SERVEURS

	Ancien palier	Palier actuel
Téléadministration		Windows: TS ou VNC (2/2004)
		UNIX/AIX/LINUX: OpenSSH 3.6 ** (2/2006) via carte à puce
		putty (11/2007)
Ordonnancement de tâches	Gepware 4.2 (2/2006)	Gepware 4.3 (2/2007)
Supervision Système		Sysload 4.8 (2/2006)
	Big Brother 1.9 (6/2005)	NAGIOS 1.2 (11/7)
Sauvegarde		Netbackup 5.0 (2/2006)
Différentiel de fichiers		KDiff3 (11/2007)

AUTRES TECHNOLOGIES ET NORMES RESEAUX

	Ancien palier	Palier actuel
Transport		TCP+UDP/IP (6/2003)
Protocole Serveurs Web		HTTP 1.1, HTTPS (6/2003)
Protocoles messagerie		SMTP/ESMTP, POP3, IMAP4 (6/2003)
Protocole transfert de fichier		FTP (6/2003)
Protocole d'interrogation d'annuaire		LDAP (6/2003)
Protocole inter-applicatif		RMI pour JAVA (9/2003)

DEVELOPPEMENT - OUTILS

	Ancien palier	Palier actuel
Standard de Modélisation		UML 1.4 (6/2003)
Outil de Modélisation		Rational Rose (6/2003)
**AGL ** **	Eclipse 3.0.2 (+ Plug-in) (2/2006)	Eclipse 3.1.2 (+ Plug-in) (2/2007)
**Gestion des sources ** **		CVS 1.11 (2/2006)
Interrogation SGBDR	SQUIRREL 1.0 ** (6/2005)	SQUIRREL 2.6.1 (11/07) TOAD
**Gestion des bogues ** **	Bugzilla 2.16.6 (6/2005)	Bugzilla 2.18 (2/2007)
**LANGAGES Serveurs ** **	Java JDK 1.4.1 (6/2003)	Java JDK 1.5 (11/2007)
		ASP (6/2003)
Design WEB		Dreamweaver (2/2004)
Tests	WinRunner, TestDirector (6/2003)	Utiliser les produits libres (11/2007)
**Tests de charge ** **		OpenSTA 1.4.3 (2/2006)
Manipulation XML		XML Spy (6/2003)
**Construction programmes JAVA ** **		Ant 1.6 (2/2007)
**Editeur Syntaxique * **		JEdit 4.2 (6/2005)
Présentation animée		TourOperator 0.3.47 (2/2006)

DEVELOPPEMENT - REGLES

Consulter le document spécifique

Ce palier de spécifications s'applique à tous les nouveaux systèmes à développer, à intégrer dans l'infrastructure de la Douane, et représente la cible de portage des applications existantes. Ancien palier désigne les versions antérieures encore installées. * indique logiciel libre, ** : support spécifique par le marché LL *Premier palier: 6/2003*

BIBLIOTHEQUES JAVA

	Ancien palier	Palier actuel
Accès aux données		JDBC (6/2003) JDO : ne pas utiliser (2/2004) Hibernate 2.1.6 (11/2007) ehcache 0.9 (11/2007)
Marshalling XML-SGBDR *	OJB (6/2005)	Ne pas utiliser OJB (11/2007)
Marshalling XML-JAVA **	CASTOR 0.9.5 (2/2007)	CASTOR 0.9.5 patché (11/2007)
Analyse XML **		Xerces 2.6 (2/2007)
Appels de fonctions distantes		SOAP (6/2003) , RMI (9/2003)
Impression légère **		FOP 0.2 + Scriptura Itext 1.3 (11/2007)
Structure pages Web **	STRUTS 1.2.4(2/2007) / Tiles	STRUTS 1.2.9(11/2007) / Tiles
Gestion Mails *		JavaMail 1.3.2 (6/2005)
Gestion traces applicatives *	Log4J 1.2.9 (6/2005)	Log4J 1.2.13 (2/2007)
Tests Unitaires **		Junit 3.8 (2/2006)
Générateur des squelettes *		JUnitDoclet (6/2005)
Gestion des listes *	DisplayTag 1.0 (2/2006)	DisplayTags 1.1.2 (11/2007)
Client FTP		Apache Common Nets 1.4 (11/2007)
WebServices		Jaxws 2.1.2 (11/2007)
schémas XML-JAVA		Jaxb 2.1.3 (11/2007)
Framework léger		spring 1.2.4 (11/2007)
Annotations/Configuration		XDoclet 1.2.3 (11/2007)

	Ancien palier	Palier actuel
REFERENTIEL DROITS D'ACCES/ UTILISATEURS		RUSH
REFERENTIEL OPERATEURS		ROSA
REFERENTIEL TARIF		RITA
Supervision *		Client Big Brother V1.0
Messagerie Applicative (avec extérieur douane)	MAREVA (11/2005)	MAREVA 2.4 (11/2007) Tradexpress (2/2007)
Commission Européenne		CCN/CSI
Charte d'ergonomie Internet		Voir sur ALADIN
Cryptographie		BouncyCastle 128 (2/2007)

Ce palier de spécifications s'applique à tous les nouveaux systèmes à développer, à intégrer dans l'infrastructure de la Douane, et représente la cible de portage des applications existantes. Ancien palier désigne les versions antérieures encore installées. * indique logiciel libre, ** : support spécifique par le marché LL *Premier palier: 6/2003*

Annexe 17 : Le palier technique de la DGDDI version 2010C

SYSTEME D'INFORMATION ET DE TELECOMMUNICATION DOUANE

DG-C2 – juin 2010 — **PALIER TECHNIQUE TOUT INTERNET – version 2010C** — http://dg-c2.douane — 1

POSTE DE TRAVAIL : tous les postes doivent passer au palier dans l'année

	Anciens paliers encore présents	Palier actuel
Système d'exploitation		XP Pro SP3 (3/2009). .Net 3.5 inclus
Environnement JAVA *		J2RE 1.6.0_07 (3/2009)
Navigateur WEB	IE 6 SP3 (3/2009) Firefox 3.0.4 (3/2009)**	IE 6 SP3 (3/2009) Firefox 3.5.3** (2/2010) +Extension Coral IE Tab v1.63.20091024 (2/2010)
Outil client de messagerie		OE 6 SP3 (3/2009) Outlook 2003 (VIP)
Bureautique **		OOo 3.0.0 (3/2009) +extension Connecteur Terminologique v1.0.0 (2/2010) formats ODF (11/2007)
		Gantt Project 2.0.10
Création PDF		PDF Creator 0.93
Visualisation	Flash Player 9.0.28.0	Flash Player 10.0.32.18 (2/2010)
	Acrobat Reader 7.0.8	Acrobat Reader 9.2 (2/2010)
	Visionneuses MS Office 2003	Visionneuses MS Office 2003/2007
		SVG Viewer 3.03 (2/2010)
Compression		Win XP (par défaut) 7ZIP4.42 (en complément)
Accès à distance	VNC 4.12 (2/2004) ne doit plus être utilisé	Bureau à distance Xp Assistance à distance Xp
		Cygwin (2009)
		InventairePDT 1 (2009)
Transferts volumineux		Grokoli version sur Aladin
Gravage		CD Burner XP Pro 4.2.5 (2/2010)

	Anciens paliers encore présents	Palier actuel
Multimédia		VLC 1.0.2
		Codecs pour WMP9:(2/2010) K-LiteCodecPack510Full
		Paint.Net 3.36 (2/2010)

COMPATIBILITE APPLICATIONS - EXPLORATEURS

	Anciens paliers encore présents	Palier actuel
Niveau pages Web		HTML 4.01 (XHTML) (6/2003)
Résolution écran	1024 x 768 pixels	1280 x 1024 pixels (3/2009)
SCRIPT sur Pages WEB		Javascript 1.5 ECMA-262 r3 (6/2003)
ACCES ALADIN (intranet)		IE 6 + Firefox 3 (4/2009)
ACCES PRODOUANE (internet) versions minimas supportées		IE 5 + Firefox 1.0 + Mozilla 1.0

TERMINAUX MOBILES (2009)

	Anciens paliers encore présents	Palier Actuel
Voix (téléphones)	Nokia 5500 Nokia E51	Nokia 1662 Nokia E51 Samsung E2100
Data exclusivement	Carte Edge Sierra 775	Carte Edge Sierra 850 Clé USB Huawei E180
Voix et data simultanément	Nokia E61	Nokia E61I

N.B. Tout ordinateur portable au palier est compatible Cristal Mobile

SECURITE POSTE CLIENT

SECURITE: La sécurité fait l'objet de documents spécifiques (PSIT et annexes).

	Ancien palier	Palier actuel
Antivirus poste client		Virascan 8.5 (1/2008)
Carte à puce		Carte CADO (2/2007) lecteur pcmcia/usb

Ce palier de spécifications s'applique à tous les nouveaux systèmes à développer, à intégrer dans l'infrastructure de la Douane, et représente la cible de portage des applications existantes. * indique logiciel libre, ** : support spécifique par le marché LL. RGI : compatibilité RGI *Premier palier: 6/2005* NOUVEAUTES EN JAUNE

CONFIGURATION MATERIELLE ORDINATEUR DE BUREAU

Modèle Tour 2010 : Acer Veriton M 480 G

Modèle Portable 2010 : Acer Travelmate PM 4730

Micro-ordinateur compatible PC, Boitier format miniTour, carte mère ATX ou BTX.

Processeur de type X86 en 32 bits. Doit permettre à la configuration d'obtenir un résultat minimum au test demandé; actuellement 110 au test Sysmark 2007.

RAM 2 GO, extensible à 4 GO avec emplacement libre

Un disque dur de capacité de 160 GO

Lecteur/graveur de DVD ROM

un clavier 105 touches AZERTY français disposant du symbole de l'euro ;

une souris USB optique, 2 boutons et une molette, cable 1,5 m+

un contrôleur Ethernet 100 Mbits/s intégré sur la carte mère

un contrôleur audio intégrés ;

un contrôleur video intégré pouvant afficher les résolutions de 1680x1050 et 1920x1080

Ports: 4 USB2.0 (hors clavier et souris), dont 2 sur la face avant, 1 PCI express 16x, 1RJ 45, 1video VGA+

1 baie 3,5 libre

Normes Niveau sonore ISO 9296 (Maximum 35dB(A) en mode d'attente, Maximum 40dB(A) lors de l'accès à un disque dur), consommation d'énergie « Energy Star »

En option: lecteur interne de cartes à puce

Moniteur plat LCD 22 pouces (résolution native minimale: 1680 x 1050).
NB : les applications intranet sont conçues et optimisées pour une fenêtre d'affichage de 1024*768 (anciennes applications) ou 1280 x 1024 (nouvelles applications).

Documentation technique en français

CONFIGURATION MATERIELLE ORDINATEUR PORTABLE

Processeur de type X86 en 32 bits. Doit permettre à la configuration d'obtenir un résultat minimum au test demandé : actuellement 90 au test Bapco Sysmark 2007

RAM 2 GO sur une seule barrette standard, extensible à 4 Go. 1 emplacement mémoire disponible.

Disque dur 80 GO

Ecran TFT 14 pouces affichant au minimum une résolution de 1024x768; format16/9 non autorisé

Poids maximum de 2,500 kg, y compris le lecteur optique et la batterie principale

Autonomie minimale de 3h00, mesurée avec Bapco Mobilemark 2007

Circuit son intégré

Clavier AZERTY français avec sérigraphie "€"

Dispositif de pointage intégré

Lecteur/graveur DVD

Circuit réseau ETHERNET 10/100, conforme aux normes en vigueur.

Un emplacement PCMCIA (PC Card) adaptable en Expresscard

Ports: 3 x USB 2.0, écran externe (pouvant afficher une résolution de 1680x1050)

Système de sécurité physique (point d'attache pour verrou)

Option : batterie supplémentaire, kit allume-cigare

Sacoche de transport

Documentation technique en français

CONFIGURATION MATERIELLE ORDINATEUR CRISTAL MOBILE DEDIE
Ces anciens postes disposaient de 2 ports (USB ou PCMCIA), pour connexion de carte Cristal Mobile et lecteur CADO.

MASTERISATION: les PC sont acquis masterisés : duplication du master douane sur les postes en usine avant livraison et fourniture d'un support (DVD) de restauration.

COMPLEMENTS-GARANTIE
Les micros et imprimantes disposent d'une garantie sur site renforcée de 3 ans (4 ans pour les UC).

SYSTEME D'INFORMATION ET DE TELECOMMUNICATION DOUANE
PALIER TECHNIQUE TOUT INTERNET – version 2010C

RESEAUX LOCAUX ET ACCES POSTES DE TRAVAIL

	Anciens paliers encore présents	Palier actuel
Equipements actifs (LAN des services)		Nortel ER8452GT-PWR / 4550 24/48 ports (6/2008)
Câblage		Catégorie 5 (6/2003) 100 Mb/s
WIFI	IEEE 802.11b (6/2003) IEEE 802.11g (2/2006)	Wifi à supprimer (3/2009) raisons de sécurité
CPL	Homeplug 1.0 (12/2006) minimum 85 Mb/s	CPL à supprimer (3/2009) (prévision TOIP)
Réseau mobile	carte EDGE (2/2006) à remplacer	Edge Sierra 850 (2/2006) via PDA E61i (2/2007) via PDA E71 (2/2010) clé USB Huawei (2/2009)
Analyse LAN **	Ethereal 0.10 (2/2006)	wireshark 0.99 (2/2007)
**		NMAP 3.7 (2/2006)
Remplacement serveurs NT		VMWare (4/2009

SERVEURS DE MESSAGERIE ET D'ANNUAIRE

	Anciens paliers encore présents	Palier actuel
SERVEURS DE MESSAGERIE Interpersonnels		Sun iPlanet V5.1 SP2 messaging server (2/2004) Exchange 2003 (VIP)
Filtrage messagerie		Trend IMSS5.7 (2/2007) Trend TMCM 3.0 (2/2007)
SERVEURS D'ANNUAIRE Interpersonnels		Sun Iplanet V5.2 directory server (2/2004)
Serveurs de messagerie* *	Postfix 2.0 (2/2006)	Postfix 2.5 (6/2010)
Serveur POP/IMAP **		CYRUS IMAP 2.2 (2/2006)
Annuaires **	LemonLDAP 1.2.4 (12/2008)	OpenLDAP 2.2 (2/2006) LemonLDAP NG 0.9.4.1 (02/2010)

SERVEURS APPLICATIFS

	Anciens paliers encore présents	Palier actuel
OS		AIX 5.2 (6/2003)
*	LINUX REDHAT AS4 (64 bits) (LARI) (11/2007)	LINUX REDHAT AS5.4 64 bits (LARI) (02/2010)
	Windows 2000 SP3 (2/2004)	Windows 2003 SPx (11/2007) Windows 2008 SPx (12/2009)
SHELL **		BASH 2.0.5 (tous OS) (2/2006)
SGBDR	Oracle 9i (6/2003) Oracle 10g (2/2006)	Oracle 11g (4/2010) PostgreSQL 8.3.x (5/2010)
	SQL Server 2000 SP3 (6/2003)	SQL Server 2005 SPx (11/2007) SQL Server 2008 SPx (12/2008)
SERVEURS WEB **	Apache 2.0 (2/2006) mod_jk1.2 mod_ssl 2.0 ATWSuite 6.4 (2/2007)	Apache 2.2 (12/2009) mod_jk1.2 mod_ssl 2.0 ATWSuite 6.7 (6/2010)
**	OpenSSL 0.9.7 (2/2007)	OpenSSL 0.9.8 (6/2010)
		IIS 5/Windows (6/2003)
WEB/Servlet **	TOMCAT 4.1 (6/2005)	TOMCAT 5.2 (11/7)
J2EE **	Weblogic 8.1 SP2 JBOSS 4.0.4 (11/7)	JBOSS EAP 5.0 (02/2010) XCommand 0.1.2 (patche Weblogic) (11/2007

Ce palier de spécifications s'applique à tous les nouveaux systèmes à développer, à intégrer dans l'infrastructure de la Douane, et représente la cible de portage des applications existantes. * indique logiciel libre, ** : support spécifique par le marché LL [308] : compatibilité RGI *Premier palier: 6/2005* *NOUVEAUTES EN JAUNE*

AUTRES FONCTIONS SERVEURS

	Anciens paliers encore présents	Palier actuel
Serveur d'émulation GCOS-IBM		ILEX (6/2003)
Infocentres	Business Objects WEBI 6.0 (6/2003) Outils ORACLE BI Standard Edition(2/2007)	Choix en cours (2009)
Serveur de temps ** RGI		Ntp 4.2 (2/2006)
Transfert de fichiers SMB* *		Samba 3.0 (2/2006)
Synchronisation **		Rsync 2.6.5 (2/2006)
Partage de Fichiers UNIX **		NFS 1.0.5 (2/2006)
Remontée et stockage d'événements **	Rrdtool 1.2.10 (2/2006)	Rrdtool 1.2.26 (6/2010)
Statistiques systèmes **		sysstat 6.0.1 (2/2006)
Récupération de pages WEB *		Wget 1.8.2 (2/2006)

ADMINISTRATION SERVEURS

	Anciens paliers encore présents	Palier actuel
Téléadministration		Windows: TS (2/2004) + SSH (2009)
		UNIX/AIX/LINUX: OpenSSH 3.6 ** (2/2006) via carte à puce
		putty (11/2007)
Ordonnancement de tâches	Gepware 4.2 (2/2006)	Gepware 4.3 (2/2007)
Supervision Système		Syslead 4.8 (2/2006)
	NAGIOS 2.9 (05/2009)	NAGIOS 2.10 (6/2010)

	Anciens paliers encore présents	Palier actuel
Sauvegarde		Netbackup 5.0 (2/2006)
Différentiel de fichiers		KDiff3 (11/2007)

AUTRES TECHNOLOGIES ET NORMES RESEAUX

	Anciens paliers encore présents	Palier actuel
Transport RGI		TCP+UDMP (6/2003)
Protocole Serveurs Web RGI		HTTP 1.1, HTTPS (6/2003)
Protocoles messagerie RGI		SMTP/ESMTP, POP3, IMAP4 (6/2003)
Protocole transfert de fichier		FTP (6/2003) SFTP + FTPS (4/2009)
Protocole d'interrogation d'annuaire RGI		LDAP V3 (6/2003)
Protocole inter-applicatif		RMI pour JAVA (9/2003)
Voix sur IP		SIP - H323 (4/2009)
Temps réel		RTP (4/2009)

Ce palier de spécifications s'applique à tous les nouveaux systèmes à développer, à intégrer dans l'infrastructure de la Douane, et représente la cible de portage des applications existantes. * indique logiciel libre, ** : support spécifique par le marché LL. RGI : compatibilité RGI *Premier palier: 6/2003* *NOUVEAUTES EN JAUNE*

DEVELOPPEMENT - OUTILS

	Anciens paliers encore présents	Palier actuel
Standard de Modélisation		UML 1.4 (6/2003)
Outil de Modélisation		Rational Rose (6/2003)
AGL **	Eclipse 3.2 (+ Plug-in) (3/2008)	Eclipse 3.4 (+ Plug-in) (12/2009)
Gestion des sources **	CVS 1.11 (2/2006)	CVS 1.12 (6/2010)
Interrogation SGBDR		SQUIRREL 1.0 ** (6/2005) TOAD
Gestion des bogues **	Bugzilla 2.16.6 (6/2005)	Bugzilla 2.18 (2/2007)
LANGAGES Serveurs **	Java JDK 1.5 .0.10 (11/2007)	Java JDK 1.6 (6/2010)
		ASP (6/2003)
Design WEB		Dreamweaver (2/2004)
Tests	WinRunner, TestDirector (6/2003)	Utiliser les produits libres (11/2007)
Tests de charge **		OpenSTA 1.4.3 (2/2006)
Manipulation XML		XML Spy (6/2003)
Construction programmes JAVA **	Ant 1.6 (2/2007)	Ant 1.7 (6/2010)
Editeur Syntaxique *		JEdit 4.2 (6/2005)
Présentation animée		ToxOperator 0.3.47 (2/2006)

DEVELOPPEMENT ET ARCHITECTURE - REGLES

Règles de programmation: document spécifique V2.2
Documents d'architecture (DAT): Document type

BIBLIOTHEQUES JAVA

	Anciens paliers encore présents	Palier actuel
Accès aux données	Hibernate 2.1.6 (11/2007) ehcache 0.9 (11/2007)	JDBC (6/2003) JDO : ne pas utiliser (2/2004) Hibernate 3.2 (6/2010) ehcache ** 1.4 (6/2008)
Marshalling XML-SGBDR *	OJB (6/2005)	Ne pas utiliser OJB (11/2007)
Marshalling XML-JAVA **	CASTOR 0.9.5 patché (11/2007)	CASTOR 1.3.1 (6/2010)
Analyse XML **		Xerces 2.6 (2/2007)
Appels de fonctions distantes		SOAP (6/2003) , RMI (9/2003)
Impression légère **	FOP 0.2 + Scriptura Rtext 1.3 (11/2007)	FOP 0.95 + Scriptura (6/2010) Itext 2.1.x (6/2010)
Structure pages Web **	STRUTS 1.2.4(2/2007) / Tiles	STRUTS 1.2.9(11/2007) / Tiles
Gestion Mails *		JavaMail 1.3.2 (6/2005)
Gestion traces applicatives *	Log4J 1.2.13 (2/2007)	Log4J 1.2.15 (6/2010)
Gestion batch multithreadé **		Quartz 1.4.2 (11/2007)
Tests Unitaires **	Junit 3.8 (2/2006)	Junit 4.4 (6/2010)
Tests et Performances		TPTP 4.5.2 (02/2010)
Générateur des squelettes *		JUnitDoclet (6/2005)
Gestion des listes *	DisplayTag 1.0 (2/2006)	DisplayTags 1.1.2 (11/2007)
Client FTP		Apache Common Net 1.4 (11/2007)
WebServices		Jaxws 2.1.2 (11/2007)
schémas XML-JAVA		Jaxb 2.1.3 (11/2007)
Framework léger	spring 2.0.4 (3/2008)	spring 2.5 (6/2010)
Annotations/Configuration		XDoclet 1.2.3 (11/2007)

Ce palier de spécifications s'applique à tous les nouveaux systèmes à développer, à intégrer dans l'infrastructure de la Douane, et représente la cible de portage des applications existantes. * indique logiciel libre, ** : support spécifique par le marché LL [RR] : compatibilité RGI *Premier palier: 6/2003* *NOUVEAUTES EN JAUNE*

	Anciens paliers encore présents	Palier actuel
REFERENTIEL DROITS D'ACCES/ UTILISATEURS		RUSH
REFERENTIEL OPERATEURS		ROSA
REFERENTIEL TARIF		RITA
Supervision *	Client Big Brother V1.0	Nagios / SAT (1/2008)
Messagerie Applicative (avec extérieur douane)	MAREVA (11/2005)	Client MAREVA 2.4.3.1 (11/2007) Tradexpress (2/2007)
Commission Européenne		CCN/CSI
Charte d'ergonomie Internet		Voir sur ALADIN
Cryptographie		BouncyCastle 128 (2/2007)

QUALIMETRE LOGICIELLE

	Anciens paliers encore présents	Palier actuel
Vérification normes de codage**		Checkstyle 4.0.1 (3/2008)
Qualité code source JAVA **	PMD 3.1.2 (3/2008)	PMD 3.2.6 (02/2010)
Analyse Statistique code JAVA**		Metrics 1.3.6 (3/2008) CAP 1.2.0 (3/2008)

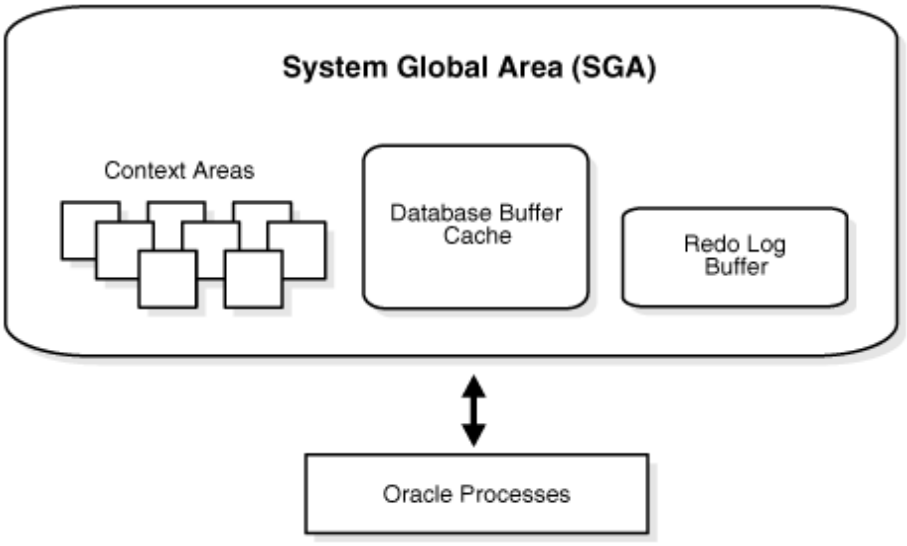
Annexe 18 : Classe DB_Conn

```
package fr.gouv.douane.morse.server;

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import oracle.jdbc.rowset.OracleCachedRowSet;

/**
 * création d'une connexion db_conn */
public abstract class DB_Conn {

	public DB_Conn() {

		// récupère la base de données Oracle
		getServerOracleOn();
	}

	private void getServerOracleOn() {
		String hostname = null;
		try {
			// Execute la commande
			String command = "hostname";
			Process child = Runtime.getRuntime().exec(command);

			// récupère et lit l'entrée
			java.io.InputStream in = child.getInputStream();
			hostname = "";
			int c;
			while ((c = in.read()) != -1) {
				hostname += (char) c;
			}

			in.close();
			child.destroy();
		}
```

```
		catch (IOException e) {
			e.printStackTrace();
		}

	}

	/** db conn */
	protected Connection getConn() {
		Connection conn = null;
		// définit une connexion à une base oracle
		String url = "jdbc:oracle:thin:@localhost:1521:XE";
		String driver = "oracle.jdbc.OracleDriver";
		String user = "morse";
		String pass = "manager";

		try {
			Class.forName(driver).newInstance();
			conn = DriverManager.getConnection(url, user, pass);
		}
		catch (Exception e) {
			// erreur
			System.err.println("SQL Connection Error: ");
			// pour débugger l'erreur
			e.printStackTrace();
		}

		if (conn == null) {
			System.out.println("~~~~~~~~~~ pas de connexion SQL");
		}

		return conn;
	}

	/* get recordset row count */
	protected static int getResultSetSize(OracleCachedRowSet resultSet) {
		int size = -1;
```

```
                try {
                        resultSet.last();
                        size = resultSet.getRow();
                        resultSet.beforeFirst();
                }
                catch (SQLException e) {
                        return size;
                }

                return size;
        }

}
```

Annexe 19 : Instance Oracle



Chaque démarrage de la base Oracle est associé à une instance Oracle. Quand une base de données est démarrée sur le serveur, Oracle alloue une zone mémoire appelée SGA (System Global Area) et démarre un ou plusieurs processus. Cette combinaison de SGA et de processus constitue une instance Oracle. La mémoire et les processus d'une instance gère les données associées de la base de données ainsi que les accès utilisateur concurrentiels.

Annexe 20 : Présentation du télétravail

Le « télétravail » désigne de manière générale toutes les formes de « travail à distance », c'est-à-dire les formes d'organisation et/ou de réalisation du travail rendues possibles hors de la classique unité de temps et de lieu, par les moyens de télécommunication et l'Internet dans le cadre d'une prestation de service ou d'une relation d'emploi contractualisée.(Wikipedia)

Il impacte tous les domaines, comme la médecine (télémédecine), l'éducation (e-learning), le commerce (e-commerce) et le développement informatique.

La circulaire du ministère du travail du 3 juillet 2009 parue en pleine pandémie grippale, incitait à recourir au télétravail si la pandémie devait prendre de l'ampleur. Cet exemple nous démontre que le télétravail même s'il est encore peu développé en France (7% de la population active contre 13% au niveau européen), est un moyen envisagé pour contrer des risques (dans cet exemple, sanitaires) mettant en péril l'économie d'un pays.

Le télétravail est encadré par l'Accord National Interprofessionnel (ANI) du 19 juillet 2005 qui définit le télétravail, ses conditions, les droits applicables, sa mise en œuvre. Cependant comme toute nouveauté, l'expérimentation et l'expérience conditionnent la création de lois adaptées. Et le décalage entre le droit et l'évolution des NTIC fait que le code du travail ne comporte aucun élément sur le télétravail. Ainsi une proposition de loi adoptée par l'Assemblée Nationale le 09 juin 2009 mais toujours pas adoptée par le Sénat à ce jour, devrait permettre d'intégrer dans le code du travail ce sujet jusqu'alors complètement ignoré.

L' ANI du 19 juillet 2009 définit le télétravailleur comme "*toute personne salariée de l'entreprise qui effectue, soit dès l'embauche, soit ultérieurement, du télétravail tel que défini ... ou dans des conditions adaptées par un accord de branche ou d'entreprise en fonction de la réalité de leur champ et précisant les catégories de salariés concernés.*"
Ainsi le télétravail implique toujours un contrat entre l'employeur et l'employé permettant à ce dernier d'être considéré comme un salarié à part entière et de bénéficier de tous les avatanges sociaux y afférant

Il faut distinguer le télétravailleur salarié du télétravailleur indépendant : le premier bénéficie d'un contrat d'embauche et dispose de toutes les ressources, du matériel adéquat à

l'accomplissement de son télétravail ainsi que de tous les avantages sociaux d'un salarié ordinaire ; le second est lié par un contrat commercial se traduisant par une prestation de service payante, il dispose de son propre matériel de télétravail entièrement à sa charge mais peut dans certains cas accéder aux ressources de l'entreprise pour laquelle le télétravail est effectué (tâches de développement, maintenance informatiques).

L'ANI 2005 et la proposition de loi sur le télétravail de 2009 ne concernent que les entreprises privées et impactent le code du travail. Or, l'administration publique n'est pas reprise dans ces textes et de ce fait ne possède pas de cadre réglementaire pour le télétravail.

ABREVIATIONS

AAU	Analyste Assistant Utilisateur
AEM	Action de l'Etat en Mer
AFNOR	Association Française de NORmalisation
AGL	Atelier de Génie Logiciel
AMOA	Assistance à Maitrise d'OuvrAge
ANI	Accord National Interprofessionnel
API	Application Programming Interface
CASE	Computer Aided Software Engineering
CDCF	Cahier Des Charges Fonctionnel
CDCT	Cahier Des Charges Technique
CGI	Common Gateway Interface
CID	Centre Informatique Douanier
CISD	Centre Inter-régionaux de Saisie des Données
CNAM	Conservatoire National des Arts et Métiers
DAO	Data Access Objet
DAU	Document Administratif Unique
DEB	Déclaration d'Echange de Biens
DELTA	Dédouanement En Ligne par Transaction Automatisée
DEST	Diplôme d'Enseignement Supérieur Technique
DGDDI	Direction Générale des Douanes et Droits Indirects
DI Rouen	Direction Inter-régionale de Rouen
DLL	Dynamic Link Library
DNSCE	Direction Nationale des Statistiques et du Commerce Extérieur
DRDNC	Direction Régionale des Douanes de Nouvelle-Calédonie
DRGC-MMN	Direction Régionale des Garde-Côtes – Manche Mer du Nord
DROP	Données de Référence Opérationnelles Partagées
DUT	Diplôme Universitaire de Technologie
EJB	Enterprise Java Beans
FAP	Fiche d'Avancement de Projet
GNC	Gouvernement de la Nouvelle-Calédonie
GWT	Gooble Web Tool kit

HTML	HyperText Mark-up Language
HTTP	HyperTexte Transfert Protocol
IDE	Integrated Development Environment
IHM	Interface Homme Machine
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JVM	Java Virtual Machine
LOB	Large Object ou BLOB (Binary Large OBject)
MOA	Maîtrise d'OuvrAge
MOE	Maîtrise d'OEuvre
MORSE	Module d'Organisation des Ressources Spécialisées et des Equipages
MVC	Model-View-Controller
MVJ	Machine Virtuelle Java
MVS/VM	Multiple Virtual Storage/Virtual Machine
NTIC	Nouvelles Technologies de l'Information et de la Communication
OEILLADES	Outils d'Envois Informatisés Lié au Logiciel d'Acquisition de Données d'Enquêtes Sphinx
ORM	Object/Relational Mapping
OSGi	Open Service Gateway initiative
PAU	Pupitreur Assistant Utilisateur
PDCA	Plan-Do-Check-Act
PDF	Portable Document Format
PMBOK	Project Management Book Of Knowledges
POJO	Plain Old Java Object
PPS	Plan Projet Succinct
PSE-CRA	Programmeur Système d'Exploitation – Concepteur Réalisateur d'Applications
PSE-ER	Programmeur Système d'Exploitation – Expert Réseaux
RAC	Référentiel pour les Applications Communautaires
RCP	Rich Client Platform
RGPP	Révision Générale des Politiques Publiques
RITA	Référentiel Intégré Tarifaire Automatisé
ROSA	Référentiel des Opérateurs et de Suivi des Agréments

RPC	Remote Procedure Call
RUSH	Référentiel des Utilisateurs, des Services et des Habilitations
SAU	Service d'Assistance Utilisateur
SGA	System Global Area
SGBD	Système de Gestion de Base de Données
SI	Système d'Information
SIRH	Système d'Information des Ressources Humaines
SNSM	Société Nationale de Sauvetage en Mer
SOFI	Système d'Ordinateurs pour le Fret International
SQL	Structured Query Langage
SSII	Sociétés de Services d'Ingénierie Informatique
SWT	Standard Widget Toolkit
TAI	Traitement Automatisé de l'Information
TESI	Trajectoire d'Evolution du Système d'Information
TMA	Tierces Maintenances Applicatives
TSAR	Traitement et Suivi de l'Assistance en Réseau
TSI	Techniciens des Systèmes d'Information
XHTML	eXtensible HyperText Mark-up Language
XML	eXtensible Mark-up Language

WEBOGRAPHIE

http://angelozerr.wordpress.com/2010/02/01/rcp_springdm_step15/

http://capirossi.org/

http://code.google.com/intl/fr/webtoolkit

http://www.csc.com

http://www.developpez.com/

http://www.douane.gouv.fr

http://eclipse.developpez.com

http://www.eclipse.org/

http://www.eclipsetotale.com/

http://www.hibernate.org/

http://www.hsc.fr

http://www.ibm.com/

http://www.idealliance.org

http://www-igm.univ-mlv.fr

http://www.inh.fr

http://www.iso.org/iso/fr

http://www.it-sudparis.eu/lor/

http://jasperforge.org/projects/jasperreports

http://www.java2s.com/

http://java.sun.com

http://www.javaplex.com/

http://jl2tho.blogspot.com/2007/09/tutorial-clientserveur-gwt-g6.html

http://www.jmdoudoux.fr/accueil_java.htm

http://www.journaldunet.com

http://legifrance.gouv.fr/

http://www.mkhelif.fr/2008/07/07/gwt-crer-un-service-rpc.html

http://www.mkyong.com/

http://moritan.developpez.com/tutoriels/java/gwt/premier/projet/#LI-B

http://msdn.microsoft.com/fr-fr/default.aspx

http://www.oracle.com

http://sourceforge.net/projects/ireport/files/

http://www.ulb.ac.be/

http://www.webopedia.com
http://www.wikipedia.org