LE *UNIFIED PROCESS* COMME METHODOLOGIE DE GESTION DE PROJET INFORMATIQUE

Eléments d'application en milieu sidérurgique

Yves Wautelet, Laurent Louvigny, Manuel Kolp
IAG- ISYS (Unité de Systèmes d'Information), Université Catholique de Louvain,
1 Place des Doyens,1438 Louvain-la-Neuve, Belgique
Contact : kolp@isys.ucl.ac.be

Résumé : Ce document présente une mise en perspective de la modélisation informatique orienté-objet de la cokerie de l'entreprise sidérurgique Carsid au cours de laquelle différents travaux d'analyse. Pour ce faire, la méthodologie informatique UP ou *Unified Process* (en français *processus unifié*) combinée au langage de modélisation UML ont été utilisés. Le UP définit un ensemble d'activités et de phases de développement (ces dernières sont composées d'itérations) pour le traitement efficace d'un projet informatique. Ce processus formalise un ensemble de concepts pertinents pour la gestion de projet comme les membres, les activités, les artéfacts et l'enchaînement d'activités. Les différentes phases d'un développement UML peuvent être représentées au moyen d'une série de diagrammes permettant de comprendre de manière visuelle les concepts définis. Tous les modèles s'enchaînent en passant de l'analyse à la conception, gagnant en complexité, dans un langage commun et unique, s'affinant au fur et à mesure pour arriver à l'élaboration finale du modèle. Les diagrammes permettront de comprendre sous différents angles la globalité du cas étudié en présentant une vue statique et dynamique de celui-ci. Chaque diagramme exprimera une partie de la structure totale, tout en étant un aspect particulier du modèle. Un modèle UML est toutefois toujours axé sur un ensemble d'hypothèses faites par l'équipe de travail et ne constitue donc qu'une vue particulière parmi d'autres possibles. L'équipe de travail doit, dés lors, être consciente que la vue adoptée est réductrice et qu'elle ne correspond pas toujours aux réalités et aux besoins du futur système informatique. Le UP fournit, à ce niveau, une méthodologie qui permet de prendre en compte plusieurs aspects différents et pertinents pour la construction de modèles au cours d'itérations successives.

1 INTRODUCTION

Les entreprises sidérurgiques belges, autrefois florissantes, n'ont cessé leur déclin depuis les années 60. Les villes de Liège, Charleroi et La Louvière disposant autrefois de mines de charbon à proximité ainsi que de voies navigables et où l'on pouvait acheminer pour un coût acceptable le minerai de fer extrait en Lorraine furent des sites de choix pour les usines sidérurgiques. Toutefois, les charbonnages ont petit à petit fermé leurs portes et le seul avantage compétitif dont pouvait disposer les sites wallons était le savoir-faire de leurs métallurgistes acquis au cours des décennies. Loin d'être négligeable, cet avantage peut toutefois difficilement compenser le fait de devoir importer le minerai de fer de sites lointains comme le Brésil, l'Afrique du Sud ou la Mauritanie.

Pour rester compétitif dans un tel contexte, les entreprises sidérurgiques wallonnes n'ont d'autre choix que de tenter d'augmenter la productivité de leurs installations.

Carsid (Carolo-Sidérurgie) est une joint venture récente issue du mouvement de concentration. Elle a été crée par les sociétés Duferco et Usinor Belgium S.A. (composition du capital de 60/40). Duferco est une entreprise italienne qui est notamment active en Belgique (à la Louvière et à Clabecq). Usinor fait partie du géant mondial de l'acier Arcelor, en partenariat avec le luxembourgeois Arbed et l'espagnol Acelaria. Duferco possède une part majoritaire dans la société Carsid.

Le projet sur la cokerie de Carsid, qui a vu le jour grâce à la collaboration entre l'entreprise Carsid et l'Unité de Systèmes d'Information (ISYS) de l'Université catholique de Louvain vise à réformer les systèmes d'information actuels de l'entreprise pour une meilleure exploitation des bases de données qu'elle possède. Les systèmes informatiques actuels sont de différentes conceptions et d'âges et ont été déployés au fur et à mesure du développement de l'informatique industrielle, la réforme de ceux-ci à pour but de rendre l'entreprise plus compétitive au niveau international. En effet, la complexité du processus de production et le nombre de machines capables de livrer des informations génèrent une abondance de données qui doivent être collectées et formatées de manière adéquate. Elles pourront ensuite s'avérer prépondérantes dans la prise de décision aussi bien au niveau de la production qu'au niveau du management.

Le présent document présente une mise en perspective par la méthodologie *Unified Process* (UP) des différents travaux de modélisation orienté-objet réalisés dans le cadre du projet Carsid. Ces différents travaux partagent un langage commun, UML et on été effectués successivement mais sur base d'hypothèses différentes. L'exposé de ces différents travaux selon la méthodologie et le formalisme du UP permet de mieux comprendre ces hypothèses et pourquoi il convient de modéliser sur base de celles-ci.

Le travail qui suit présente, dans sa première partie, les concepts théoriques de la gestion de projet. Tout d'abord les meilleures pratiques d'élaboration de logiciel sont exposées. Ensuite, la méthodologie UP qui exploite ces pratiques est étudiée. Pour ce faire, son architecture bidimensionnelle est présentée sous forme de schéma synthétique. Les activités et les phases (ces dernières sont composées des itérations) qui constituent ces deux axes sont également explicitées.

Dans sa seconde partie, ce travail expose l'application de la méthodologie UP à la cokerie de Carsid. Tout d'abord les concepts théoriques du UP sont appliqués à la présente étude de cas, ensuite les différentes activités d'une part et phases et itérations d'autre part menées jusqu'à présent sont expliquées et comparées.

2 ELEMENTS DE GESTION DE PROJET

La croissance en nombre, en taille et en complexité des logiciels, combinée à la nécessité d'obtenir sans cesse de nouveaux gains de productivité pousse le monde du développement de logiciels à l'utilisation de nouvelles méthodologies, non seulement au niveau de la modélisation, mais également au niveau de la gestion de projets. Le *Unified Process* ou *UP* est l'une de ces méthodologies.

2.1 Meilleures pratiques d'élaboration de logiciels

Les logiciels jouent un rôle central dans la gestion des entreprises modernes, dans les lois gouvernementales et dans la société de l'information. Ils ont permis de créer de l'information, d'y accéder et de la visualiser sous une forme auparavant inconcevable. Les logiciels sont devenus incontournables ; ils ont favorisé la croissance de l'économie mondiale et sont devenus indispensables à notre monde moderne.
Dans ce contexte, les logiciels ont considérablement crû et ce sous divers aspects : croissance en nombre, en taille et en complexité. De plus, les entreprises sont toujours à la recherche d'une augmentation de la productivité.

Pour toutes ces raisons, la création et la maintenance des logiciels est devenue de plus en plus compliquée et la création de logiciels de qualité dans les règles de l'art l'est encore plus. Par conséquent, trop de projets échouent.

Pour lutter contre l'échec, de meilleures pratiques dans l'élaboration de logiciels ont été développées. Parmi ces pratiques, on trouve :

- le développement itératif ;
- la gestion des besoins ;
- l'utilisation d'architectures basées sur les composants ;
- la modélisation visuelle ;
- la vérification continuelle de la qualité des logiciels ;
- le contrôle des changements des logiciels.

Développement itératif. Le processus de développement de logiciels classique suit le cycle de vie en cascade. Dans ce type d'approche, le développement suit un processus linéaire commençant par l'analyse des besoins puis le design, ensuite le développement, suivi des tests

unitaires et des tests systèmes. L'inconvénient de cette approche est que le risque augmente au cours du temps de sorte qu'il est coûteux de réparer les erreurs des premières phases.

Une alternative au processus à cycle de vie en cascade est le processus itératif et incrémental. Dans cette approche, l'identification des risques liés à un projet est forcée très tôt dans son cycle de vie, lorsqu'il est encore possible de réagir d'une manière rapide et efficiente.

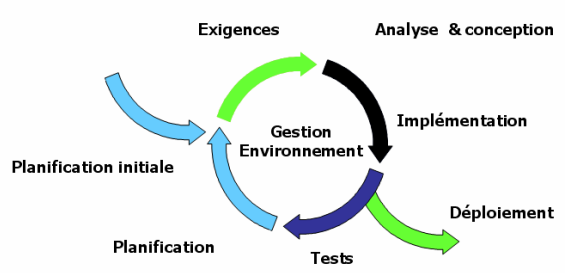

Figure 1 : Un processus itératif et incrémental. Chaque itération a pour finalité une version exécutable.

Avantages du développement itératif :

- les risques sont évalués au départ et non au cours du projet ;
- les premières itérations permettent d'avoir un feed-back des utilisateurs ;
- les tests et l'intégration se font de manière continue ;
- les jalons permettent de fixer des objectifs ;
- les avancées sont évaluées au fur et à mesure de l'implémentation ;
- des maquettes intermédiaires peuvent être mises sur pieds.

Gestion des besoins. Un besoin est une condition ou une possibilité que le système doit rencontrer. Le problème avec la gestion des besoins est qu'ils sont dynamiques : ils évoluent au cours de la vie du projet.
En effet, les utilisateurs ne savent pas au départ quelles sont leurs exigences et comment les spécifier précisément. Leurs exigences changent lorsqu'ils voient le travail accompli. Barry Boehm, professeur à l'université de Californie du Sud, appelle ce phénomène l'effet *IKIWISI* pour *I Know When I See It*.
L'identification des besoins réels est dès lors un processus continu.

Utilisation d'architectures basées sur les composants. Visualiser, spécifier, construire et documenter un système demande que celui-ci soit vu selon un certain nombre de perspectives. Toutes les personnes impliquées (utilisateurs, analystes, développeurs, intégrateurs système, testeurs, managers, etc.) apportent un agenda différent au projet, et chacun regarde le système d'une façon propre à divers moments de la vie du projet. Une architecture du système est la chose la plus importante qui puisse être utilisée pour gérer ces différents points de vue et contrôler le développement itératif et incrémental du système au cours de sa vie.

Modélisation visuelle du logiciel. L'utilisation d'un langage de modélisation standard comme UML permet aux membres de l'équipe de développement de communiquer leurs décisions l'un à l'autre sans ambiguïté.
L'utilisation d'outils de modélisation visuelle facilite la gestion de ces modèles et contribue à maintenir la consistance entre les artefacts du système : ses besoins, designs et implémentations. En résumé, la modélisation visuelle permet de gérer la complexité des logiciels.

Vérifier continuellement la qualité logicielle. Les logiciels sont 100 à 1000 fois plus coûteux à corriger après leur déploiement qu'avant. Il est donc important d'évaluer continuellement la qualité d'un système du point de vue de la fonctionnalité, de la fiabilité et de la performance.

Vérifier la fonctionnalité implique de tester chaque scénario possible, un scénario représentant un aspect du comportement du système désiré.

Contrôler les changements logiciels. L'une des grosses difficultés dans le développement logiciel est que plusieurs développeurs organisés en équipes pouvant être sur différents sites travaillent ensemble sur plusieurs itérations, versions, produits, plates-formes. En l'absence d'un contrôle discipliné, le processus de développement peut dégénérer en chaos.
Coordonner les activités et les artefacts des développeurs et des équipes implique d'établir des enchaînements d'activités (*workflows*) pour gérer les changements dans les logiciels et les artefacts. Cette coordination permet une meilleure allocation des ressources basée sur les priorités et les risques du projet et permet de gérer le travail sur ces changements au travers des itérations. Combinée au développement itératif, cette technique permet de contrôler les changements de sorte qu'il soit possible de découvrir activement les problèmes et d'y réagir.

2.2 UP

UP. Le *UP* ou *Unified Process* est un processus de conception/développement de logiciel. Il apporte une approche disciplinée pour assigner des tâches et des responsabilités dans une organisation de développement. Son but est d'assurer la production de logiciels de qualité qui rencontrent les besoins de ses utilisateurs finaux dans un horaire et un budget prédictibles.

Le UP est bâti sur ces six meilleures pratiques d'élaboration des logiciels pour délivrer un processus bien défini dans une forme adéquate pour un grand nombre de projets et d'organisations.

Processus. Un processus décrit qui fait quoi, comment et quand. Le UP définit ainsi quatre éléments primaires de modélisation :

- Le membre est le qui : le chef de projet, l'analyste, le testeur, l'utilisateur, etc.
- L'activité est le comment : analyse des cas d'utilisation, conception de cas d'utilisation, etc.
- L'artefact est le quoi : un document de l'architecture, un modèle des cas d'utilisation, un fichier exécutable, etc.
- L'enchaînement d'activités est le quand : modélisation métier, implémentation, test, etc.

Membre (ou workers ou roles). Le concept central dans un processus est celui de membre. Un membre définit le comportement et les responsabilités d'un individu ou d'un groupe d'individus travaillant de concert comme une équipe. Le comportement est exprimé en terme d'activité que le membre effectue et chaque membre est associé à une gamme d'activités. Les responsabilités du membre sont souvent exprimées en relation avec les artefacts qu'il crée, modifie ou contrôle.
Le membre doit être considéré en terme de « casquette », c'est-à-dire en fonction du rôle qui définit comment celui-ci devrait faire le travail. Par exemple, un analyste système est un individu agissant comme un analyste système qui dirige et coordonne les besoins et la modélisation par cas d'utilisation en définissant la fonctionnalité du système et en délimitant celui-ci.

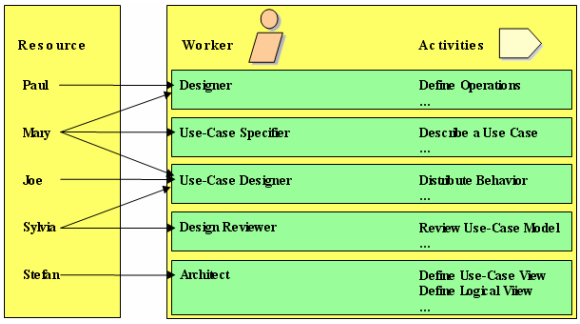

Figure 2 : Chaque membre est considéré comme un acteur

Le UP permet d'accroître la productivité en conception/développement en fournissant un ensemble d'outils communs à chaque membre impliqué dans le projet. Tous les membres partagent ainsi :

- des bases de connaissance ;
- une méthode commune ;
- une même organisation du travail ;
- un langage.

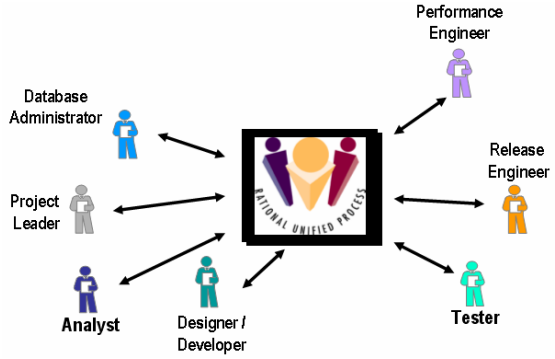

Figure 3 : Le UP comme langage commun de l'organisation

Activités. Les membres ont des activités qui définissent le travail qu'ils effectuent. Une activité est une unité de travail qu'un individu dans un rôle bien précis peut effectuer et qui produit un résultat sensé dans le cadre du projet. L'activité a un but clairement établi, généralement exprimé en terme de création ou de mise à jour d'artefacts, comme un modèle, une classe, ou un plan. Chaque activité est assignée à un membre spécifique.

Artefacts. Les activités ont des artefacts comme input et comme output. Un artefact est une patrie d'information qui est créée, modifiée ou utilisée par un processus. Les artefacts sont les produits tangibles du projet : les choses qui sont produites par le projet ou utilisées pour travailler sur le produit final. Les artefacts sont utilisés comme input par les membres pour effectuer une activité et sont le résultat ou l'output de telles activités. En vision orientée objet, les activités sont les opérations sur un objet actif (le membre), les artefacts sont les paramètres de ces activités.

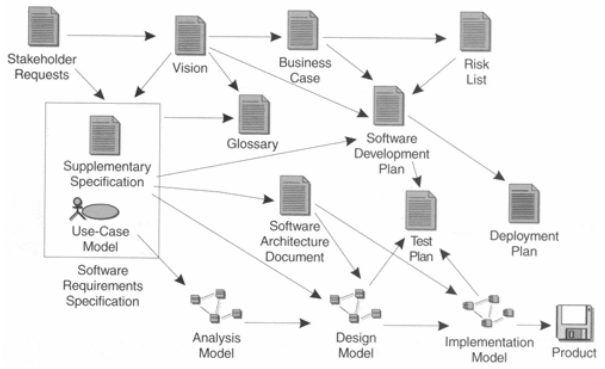

Figure 4 : Principaux artefacts du UP

Enchaînement d'activités (*workflow*). Une énumération de tous les membres, activités et artefacts ne constitue pas un processus. Il est nécessaire d'avoir une façon de décrire des séquences d'activités sensées qui produisent un résultat de qualité et montre l'interaction entre les membres. Le *workflow* est une séquence d'activités qui produit un résultat de valeur. En UML, il peut être exprimé par un diagramme de séquence, un diagramme de collaboration ou un diagramme d'activité.

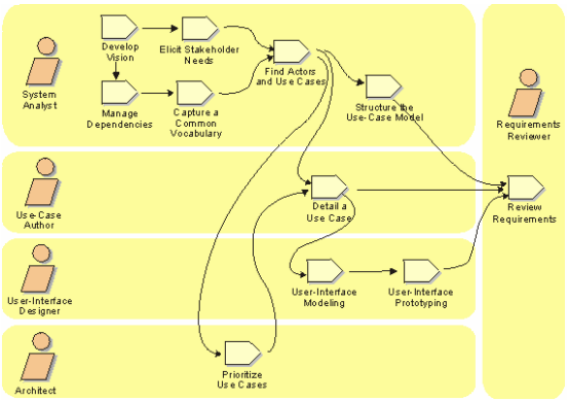

Figure 5 : Exemple de *workflow*

Architecture bidimensionnelle du UP.

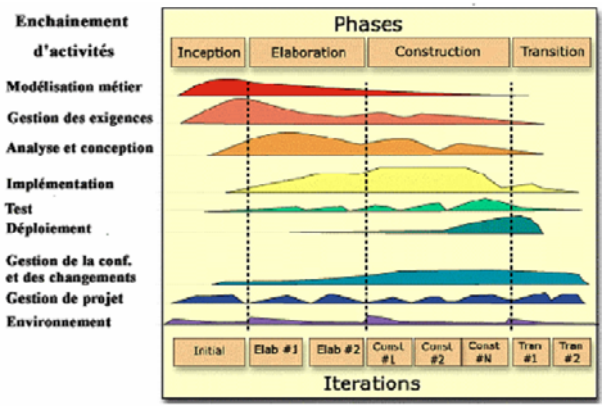

Figure 6 : Architecture bidimensionnelle du UP.

Axe vertical : activités essentielles et de soutien. Le UP possède six enchaînements d'activités essentielles ou d'ingénierie et trois enchaînements d'activités de soutien.
Les six activités essentielles sont : la modélisation métier, la gestion des exigences, l'analyse et la conception, l'implémentation, le test et le déploiement. Les trois activités de soutien sont la gestion de la configuration et des changements, la gestion de projets et l'environnement.

Modélisation métier.
La modélisation métier a pour but :

- de décrire la structure et la dynamique de l'organisation dans laquelle le système est déployé ;
- de comprendre les problèmes courants dans l'organisation et d'identifier les améliorations potentielles ;
- de garantir que les clients, les utilisateurs finaux et les développeurs partagent une vision commune de l'organisation ;
- de réaliser une base d'informations qui contiendra le cahier des charges du produit et la planification des tâches de l'organisation.

Pour atteindre ces buts, la modélisation métier décrit comment développer une vision de la nouvelle organisation et, basé sur cette vision, définit les processus, rôles et responsabilités de cette organisation dans le modèle de l'entreprise.

Gestion des exigences.
La gestion des exigences a pour but :

- d'établir et de maintenir les accords avec les clients et autres stakeholders sur ce que le système doit faire ;
- de procurer aux développeurs une meilleure compréhension des besoins du système ;
- de définir les limites du système ;
- de procurer une base pour planifier le contenu technique des itérations ;
- de procurer une base pour estimer le coût et le temps pour développer le système ;
- de définir et de construire une maquette de l'interface utilisateur basée sur les exigences et les buts de celui-ci.

Pour atteindre ces buts, la gestion des exigences décrit comment définir une vision du système et traduire la vision en un modèle des cas d'utilisation accompagné de spécifications externes constituant le cahier des charges logicielles. De plus, la gestion des exigences décrit comment utiliser les attributs des exigences pour aider à gérer la portée et le changement d'exigences du système.

Analyse et conception
L'analyse et la conception ont pour but :

- de comprendre le cahier des charges et d 'écrire les spécifications internes qui décrivent comment implémenter le système (exigences transformées dans la meilleure stratégie d'implémentation possible). L'analyse permet d'obtenir une vue interne idéale du système ;
- la conception a pour but de définir une architecture robuste du système (c'est-à-dire facile à comprendre, construire et faire évoluer) qui recouvre entièrement les besoins de celui-ci ;
- l'analyse se concentre sur le "quoi faire", la conception se concentre sur le "comment le faire".

Implémentation.
L'implémentation a pour but :

- de définir l'organisation du code en terme de sous-systèmes d'implémentation organisés en couches ;
- d'implémenter classes et objets en terme de composants ;
- de tester les composants développés comme des unités ;
- d'intégrer dans un système exécutable les résultats produits par des programmeurs individuels ou des équipes.

Test.
La phase de test a pour objectif d'évaluer le niveau de qualité atteint par le produit et d'en tirer les conclusions. Ceci ne comprend pas uniquement le produit fini, mais commence tôt dans le projet avec la validation de l'architecture et continue à travers la validation du produit fini au consommateur.
Les tests comprennent :

- la vérification des interactions des composants ;
- la vérification de la bonne intégration des composants ;
- la vérification que toutes les exigences ont été implémentées correctement ;
- l'identification et la vérification que toutes les défectuosités découvertes sont corrigées avant le déploiement du logiciel.

Déploiement.
Le but des activités de déploiement est de livrer le produit aux utilisateurs finaux.

Gestion de la configuration et des changements.
Le but de la gestion de la configuration et des changements est de tracer et de maintenir l'intégrité de l'évolution des spécificités du projet. Pendant le développement du projet, de nombreux artefacts de valeurs sont créés et représentent un investissement significatif. Ces artefacts évoluent au cours du projet et les membres doivent être capables de les identifier et de les localiser, d'en sélectionner la version appropriée, et d'examiner leur historique et la raison pour laquelle ils ont changé et d'en identifier le responsable. De même, l'équipe en charge du projet doit être à même de tracer l'évolution du produit, de capturer et gérer les demandes de changement peu importe d'où elles viennent et implémenter ces changements de façon consistante. Enfin, pour supporter l'enchaînement d'activités, il faut procurer l'information du statut des artefacts clés du projet et rassembler les mesures liées aux changements qu'ils subissent.
En résumé, le but de la gestion de la configuration et des changements est de garder la trace de tous les éléments tangibles qui participent au développement et de suivre leur évolution.

Gestion de projet.
La gestion d'un projet logiciel est l'art de mesurer les objectifs compétitifs, de gérer le risque et les

contraintes pour fournir un produit qui rencontre les besoins des consommateurs et des utilisateurs.

La gestion de projet a les buts suivants :

- procurer un cadre pour gérer les projets logiciels ;
- procurer des lignes de conduites pratiques pour planifier, répartir le personnel, exécuter et contrôler les projets ;
- procurer un dispositif de gestion du risque.

L'environnement.
Le but de l'environnement est de supporter l'organisation du développement avec les processus et les outils. Ce support inclut les éléments suivants :

- la sélection des outils de travail qui aident à réaliser les activités et leur acquisition ;
- la mise en place et la configuration des outils convenant à l'organisation ;
- la configuration du processus de développement adapté au projet ;
- le développement de ce processus ;
- les services techniques pour supporter le processus : l'infrastructure de la technologie de l'information, l'administration comptable, les sauvegardes, etc.

Axe horizontal : phases du UP.

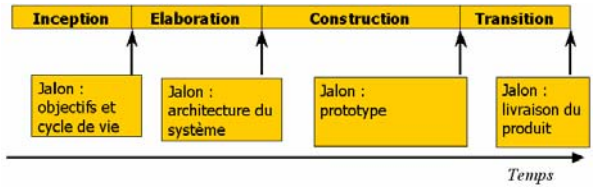

Figure 7 : phases du UP

Inception.
La phase d'*inception* a pour but de décrire la vision du produit final, de réaliser une étude de rentabilité et de risques et de définir le projet.
Les activités principales de cette phase sont :

- formuler l'ampleur du projet, c'est-à-dire capturer le contexte, les exigences et les contraintes les plus importantes de façon à pouvoir définir des critères acceptables pour le produit fini ;
- planifier et préparer un *business case* et évaluer les alternatives du point du vue du risque, du staff, de la planification, du coût, du profit, …
- synthétiser une architecture et évaluer les coûts, plans et ressources nécessaires.

La phase d'*inception* se termine par le jalon « *objectifs et cycle de vie* ».

Elaboration.
La phase d'élaboration a pour but d'analyser le domaine du problème, de construire l'architecture de base, de résoudre les éléments à haut risque et de définir la plupart des exigences. Il s'agit de la phase la plus critique du projet. A la fin de celle-ci, le travail d'engineering est terminé et l'on peut prendre la décision d'accomplir les phases de construction et de transition du projet.

Les activités principales de cette phase sont :

- élaborer une vision des cas d'utilisation les plus critiques ;
- le processus, l'infrastructure et l'environnement de développement sont élaborés et le processus, les outils et le support mis en place ;
- l'architecture est élaborée et les composants sont sélectionnés et intégrés dans un scénario primaire. Les leçons tirées de ces activités peuvent résulter en un changement d'architecture.

La phase d'élaboration se termine par le jalon « *architecture du système* ».

Construction.
Il s'agit de construire les fonctionnalités qui ne font pas partie de l'architecture de base et préparer le déploiement.

Les activités principales de cette phase sont :

- la gestion de ressources, le contrôle de ressources et l'optimisation de processus ;
- le développement complet de composantes et le test par rapport au critères d'évaluation définis ;
- l'appréciation des produits par rapport aux visions définies.

La phase de construction se termine par le jalon « *prototype* ».

Transition.
Le but de la phase de transition est de procurer le logiciel à la communauté d'utilisateurs. Après que le produit ait été donné à l'utilisateur, il est généralement nécessaire de développer de nouvelles versions, de corriger certains problèmes ou de finir certaines choses postposées.

Il s'agit d'effectuer les bêtas tests pour valider le nouveau système auprès des utilisateurs et le déploiement de celui-ci.

La phase de transition se termine par le jalon « *livraison du produit* » ou par une nouvelle itération.

3 DESCRIPTION DU PROCESSUS DE COKÉFACTION

Le travail de préparation du coke à partir d'un charbon adéquat nécessite un long et lourd processus de transformations successives du charbon (figure 2). Il s'agit de produire un coke de bonne qualité tout en traitant tous les co-produits générés aux cours des différentes étapes. La cokerie est divisée en trois secteurs.

Secteur préparation. La réception et le traitement primaire du charbon cokéfiable constituent le début de la chaîne dans les installations de cokéfaction.
Les réceptions de charbon transitent par les ports d'Anvers et de Rotterdam en provenance des États-Unis, de Pologne, d'Afrique du sud et d'Australie. Les caractéristiques propres des charbons diffèrent selon leurs origines. Le transfert depuis les ports a lieu selon trois moyens de transport : par route, par eau et par voie ferroviaire, cette dernière manière étant la plus régulièrement utilisée à la cokerie de Carsid.
Les charbons qui arrivent sont stockés soit dans une fosse soit dans une réserve qui sert de stock de sécurité afin de protéger l'entreprise des aléas éventuels des fournisseurs ou des livreurs. La production réalisable au moyen de ce stock est d'environ 15 jours au maximum. Le stockage prolongé pourrait en effet nuire à la qualité du charbon utilisé, ce qui se répercuterait inévitablement sur la qualité du coke produit.
Une série d'instruments de transports (chariots racleurs, bandes transporteuses,…) sont utilisés pour le transport de charbon depuis la réserve vers des silos de stockage. Ce transport est surveillé par une machine qui détecte et capte les métaux qui seraient éventuellement présents afin d'éviter les déchirures de la bande. Les silos sont divisés en deux catégories selon qu'ils soient silos de réserve ou doseurs. Les silos doseurs ont une capacité de 250 tonnes et sont au nombre de 10. Ils permettent de mélanger les charbons de différentes origines afin d'obtenir les caractéristiques chimiques désirées. Ce mélange se fait en dessous des silos : les quantités de charbon désirées sont déversées depuis chaque silo via 10 chariots racleurs sur une bande de transport unique qui conduit le mélange vers le broyage.
Les silos de réserve ont une capacité dix fois plus grande que les silos doseurs et permettent de remplir ces derniers en cas de besoin.
Le premier traitement agissant sur la nature physique du charbon débute lors du broyage du lot transporté à la sortie des silos doseurs. Après un passage par une série de bandes transporteuses, un broyeur va agir sur les différents charbons présents dans le mélange afin d'obtenir une granulométrie homogène. Celle-ci diffère en effet en fonction de l'origine des charbons. Un détecteur de métaux identique à celui du transport de charbon brut contrôle les opérations afin de protéger la machinerie. A l'issue du broyage, le produit obtenu a pour dénomination « pâte à coke ». Elle est transférée au moyen d'une série de transporteurs de la tour de broyage vers la tour à charbon. Cette tour est située au dessus des fours et possède une capacité de 3.600 tonnes, soit les besoins équivalent à une journée de production.

Secteur fours. Avant de débuter l'opération de distillation de la pâte à coke, il faut transférer la pâte dans les fours. L'équipe chargée des fours transfère à cet effet une quantité de pâte dans une machine appelée enfourneuse. Cette machine, située au dessus des batteries, remplit ensuite les fours par des ouvertures situées au dessus de ceux-ci, les bouches d'enfournement. Ces ouvertures se présentent sous la forme de bouchons et au nombre de quatre par four. Durant le remplissage, une autre machine appelée défourneuse s'occupe elle d'égaliser la charge de pâte tout le long du four afin d'assurer une meilleure répartition de la charge et homogénéiser la cuisson. Les remplissages, ou enfournements, suivent un planning précis qui tend à maximiser le nombre d'enfournements journaliers.
Le remplissage fini, les ouvertures supérieures et latérale sont fermées et le processus de cuisson de la pâte débute. L'opération dure de 16 à 19 heures, selon la batterie dans laquelle se situe le four. Cependant, il est difficile d'évaluer les durées exactes de cuisson à cause de nombreuses incertitudes liées aux fours. Les enfournements sont effectués selon un pas de cinq, ce qui signifie que la procédure est d'enfourner le four (n+5) après avoir enfourné le four (n). Cette méthode permet d'éviter de trop grandes variations de température dans les fours.
Les fours sont rassemblés par série dans des ensembles appelés batteries. Celles-ci sont au nombre de quatre et composées de 20 à 50 fours, ce qui donne un ensemble total de 122 fours pour la cokerie de Carsid. Les propriétés des fours au niveau des temps de cuisson théoriques et des quantités

enfournables sont différentes selon la batterie à laquelle ils appartiennent. Les quatre batteries sont alignées en ligne avec la tour à charbon située au milieu. Cette disposition permet une circulation plus facile de la machinerie autour des fours.
La composition en batterie trouve son origine dans une volonté de réduire les pertes calorifiques dans chaque four. En effet, l'espace entre les fours, appelé piedroit, est composé d'une série de cheminées verticales par lesquelles passent les gaz permettant la cuisson dans les fours. Ces cheminées, appelées des carneaux, contribuent donc à la cuisson de deux fours contigus. L'échange de calories se fait par conductivité sans contact direct avec la pâte à coke. Les gaz utilisés sont soit issus de la cokerie soit du haut fourneau, ce qui permet de rendre la cokerie indépendante au niveau énergétique. Les fumées de combustion sont récupérées en passant dans les empilages des briques réfractaires lors d'un premier cycle. Lors d'un second cycle, les gaz de combustion les parcourent afin de récupérer leur chaleur. Le passage d'un cycle à l'autre est dénommé « inversion ». Ce passage a lieu toutes les demi heures.

Le four atteint une température de plus de 1200°C durant la cuisson, ce qui permet la transformation de la pâte en saumon de coke, coke à très haute température possédant déjà les caractéristiques finales requises. Les gaz issus de la distillation sont récupérés en vue de leur valorisation.

Secteur refroidissement. A la fin de la cuisson, l'opération de défournement, c'est à dire de vidage du four, débute. Pour l'effectuer, le four est ouvert par ses portes latérales. D'un côté, la défourneuse expulse la charge du four en la poussant. De l'autre, le guide coke, un couloir muni d'une auge mobile, réceptionne la charge et la transfère jusqu'à un wagon, le coke car. L'opération de refroidissement débute. Le coke car est conduit vers la tour d'extinction. Une quantité d'eau de 25m³ y est déversée. Elle doit refroidir et éteindre le saumon de coke. Une grande partie de cette eau s'évapore directement au contact du saumon. Les eaux résiduelles sont traitées dans un bassin de décantation afin de les réutiliser ultérieurement. Le coke est éventuellement arrosé manuellement dans le cas où il ne serait que partiellement éteint.

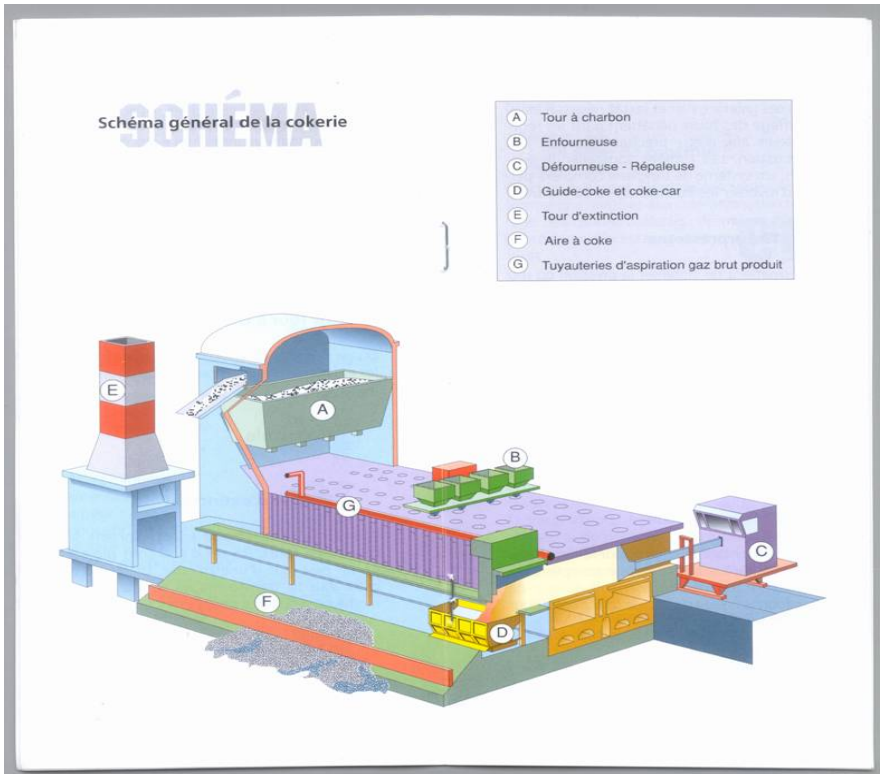

Figure 8 : Schéma général de la cokerie: enfournement, défournement et refroidissement

L'étape suivante est la mise en attente du coke, qui transite par un quai d'étalement afin de perdre son humidité. Après une période d'attente d'environ trente minutes, le coke car transférera le coke obtenu vers la tour à gros coke. Là, l'opération de criblage débute. Il s'agira de séparer les fragments que l'on considère comme « coke sidérurgique », c'est-à-dire propres à être utilisés dans l'élaboration de l'acier, des autres, valorisables à d'autres usages. La base de cette sélection se fait sur la taille des morceaux, qui doit être suffisante pour garder une bonne perméabilité de la charge. Les fragments considérés comme sidérurgiques sont expédiés vers le haut fourneau tandis que le « petit coke » est dirigé vers l'agglomération.
Certaines tâches de réparation doivent être effectuées sur les fours. Les équipes de maçons ont la responsabilité de réparer les fissures et autres problèmes pouvant survenir au niveau des fours. Ceux-ci s'usent et peuvent connaître des problèmes au niveau de leurs briques de revêtement. La réparation d'un four se fait à la température de 700°C. Il est impossible de descendre en dessous de ces températures car les briques changeraient d'état et deviendraient cassantes comme du verre. Les opérations de réparation sont donc des travaux lourds et difficiles, compte tenu des conditions de température présentes.

Il existe finalement une équipe de mesure et de réglage qui a pour but de veiller au bon fonctionnement de l'appareil de production de la cokerie. Elle contrôle la température des fours en mesurant la chaleur dans les carneaux des fours et peut donc réguler cette dernière. Elle est aussi responsable de l'inversion du cycle de chauffe de la batterie, qui a lieu toutes les demi heures environ.

4 APPLICATION DU UP A LA COKERIE

Ce chapitre aborde l'application de la méthodologie UP/UML au projet de modernisation de l'informatique de la cokerie de Carsid.

4.1 Le UP comme gestionnaire de projet

Le schéma de la figure 6 présente l'architecture bidimensionnelle du UP. La transposition de ce schéma générique aux activités effectuées dans le cadre du projet Carsid est présenté dans le schéma de la figure 9.

Notons tout d'abord que l'état d'avancement du projet Carsid ne nous a permis de travailler, au niveau de l'architecture verticale, que sur les trois premières activités de l'enchaînement défini par le UP : *la modélisation métier*, *la gestion des exigences* et *l'analyse et la conception*.

Au niveau de l'architecture horizontale du UP, seules les phases d'*inception* et d'*élaboration* ont pu être abordées.

Membres.
La phase d'*inception* constitue le travail préliminaire réalisé par Aymeric Donnay et l'équipe de chercheurs du département ISYS. Il comprend une modélisation UML complète de la cokerie.
La phase d'*élaboration* constitue le travail réalisé après l'itération préliminaire. Il comprend la rétro ingénierie des bases de données réalisée par Marti Ibarz et la modélisation UML réalisée par Laurent Louvigny et Yves Wautelet.

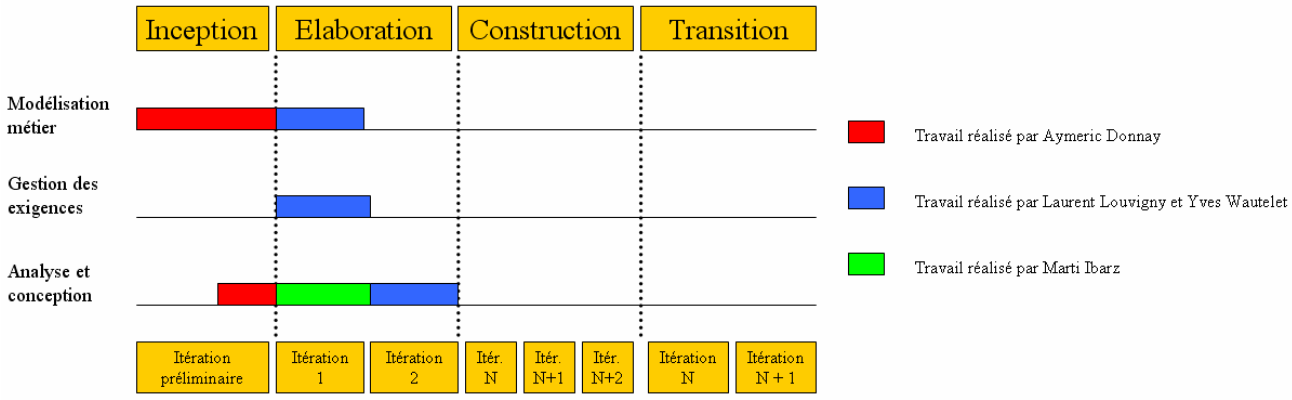

Figure 9 : schéma générique du RUP appliqué à la cokerie de Carsid.

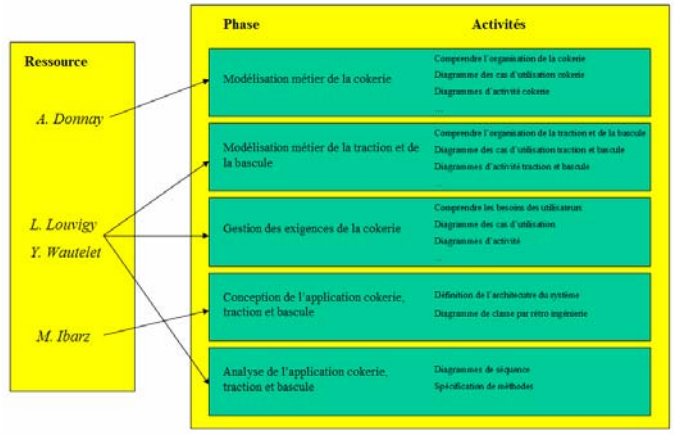

Figure 10 : Les membres du projet Carsid.

4.2 Axe vertical : les activités

Il convient maintenant de décrire plus amplement les activités réalisées dans le cadre de la modélisation métier, de la gestion des exigences et de l'analyse et la conception, soit suivant l'architecture verticale du UP.

Modélisation métier.
Pour traiter la modélisation métier de la cokerie, de la traction et de la bascule, un diagramme des cas d'utilisation ainsi que des diagrammes d'activités ont été réalisés. Ceux-ci avaient pour but de décrire la structure et la dynamique de l'organisation. Pour ce faire, les différents membres impliqués ont effectué des visites des installations concernées et interviewé des travailleurs en charge de ces installations et des responsables du système informatique actuel. Par ailleurs, ils ont également étudié les documents internes sur le processus de production fournis par l'entreprise. Ceci a permis à l'équipe comprendre l'organisation de la cokerie et de percevoir la complexité des processus à traiter avant de les modéliser. Le modèle créé permet aux membres du projet et au département informatique de Carsid de partager une vision commune de l'organisation.

Gestion des exigences.
Pour traiter la gestion des exigences de la cokerie, un diagramme des cas d'utilisation ainsi que des diagrammes d'activités ont également été réalisés. La démarche fut toutefois quelque peu différente. Les membres en charge de cette étape se sont intéressés d'une part aux machines capables de fournir des informations au système informatique et de l'utilisation possible du traitement de ces informations dans le processus décisionnel et, d'autre part, aux besoins des utilisateurs du système informatique actuel. Ces deux points de vue ont permis de définir les besoins réels en matière de traitement de l'information qui ont servi de base à la phase d'analyse du système.

Analyse et conception.
La finalité de la phase d'analyse et conception fut la mise en place d'un diagramme de classe accompagné de ses méthodes. L'architecture de ce diagramme fut dictée par la réutilisation du système de base de données relationnelles existant et ainsi définie par rétro-ingénierie de celui-ci. La spécification des méthodes présentes dans ce diagramme est le résultat de la phase de gestion des exigences pour la cokerie et de la phase de modélisation métier pour la traction et la bascule. Ces spécifications sont un contrat entre les programmeurs de la méthode et ses utilisateurs. Dans le cas présent elles serviront aux membres chargés d'implémenter le système car elles spécifient l'état du système à l'entrée de la méthode et à la sortie. Ceci sera rappelé dans le chapitre 10.

Implémentation, test et déploiement.
Ces phases du projet n'ont pas encore été réalisées.

Artefacts. Les activités ont des artefacts comme input et comme output. Ceux-ci peuvent se résumer par le tableau de la figure 10 et leur enchaînement peut se visualiser par le schéma de la figure 11.

Activité	*Input*	*Output*
Modélisation métier traction	Interviews réalisés auprès du personnel. Visite des installations. Interviews réalisés auprès des responsables informatiques.	Diagramme des cas d'utilisation. Diagrammes d'activités.
Modélisation métier bascule	Interviews réalisés auprès du personnel. Visite des installations. Interviews réalisés auprès des responsables informatiques.	Diagramme des cas d'utilisation. Diagrammes d'activités.
Modélisation métier cokerie	Interviews réalisés auprès du personnel. Visite des installations. Documents internes sur le processus de production. Interviews réalisés auprès des responsables informatiques.	Diagramme des cas d'utilisation. Diagrammes d'activités.
Gestion des exigences cokerie	Modélisation métier de la cokerie. Documents internes sur le système d'information. Interviews réalisés auprès des responsables informatiques.	Diagramme des cas d'utilisation. Diagrammes d'activités.
Analyse et conception	Modélisation métier traction. Modélisation métier bascule. Gestion des exigences cokerie. Bases de données relationnelles existantes.	Diagrammes de séquence. Spécification des méthodes. Diagrammes de classe.

Figure 11 : Artefacts comme *input* et *output*.

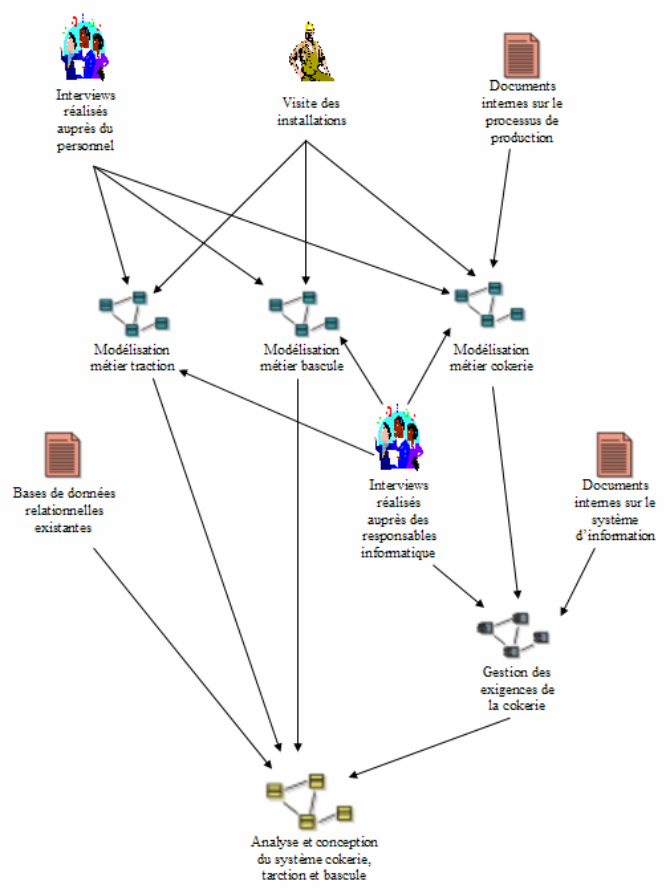

Figure 12 : Visualisation des artefacts.

Enchaînement d'activités (ou *workflow*).
L'enchaînement d'activités telles qu'il a été réalisé au cours de l'élaboration du projet peut se visualiser dans le schéma de la figure 13. Outre la séquence d'activités, on peut également y voir l'interaction entre les membres.

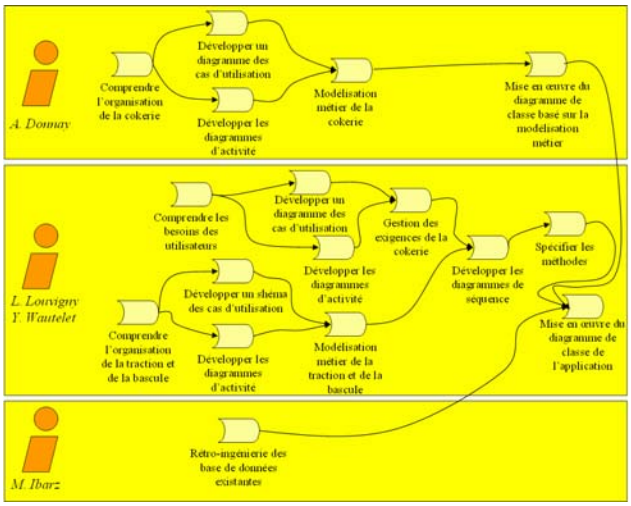

Figure 13 : Enchaînement d'activités ou workflow

4.3 Axe horizontal : phases et itérations

Inception : itération préliminaire. Lors de l'itération préliminaire, soit la phase d'inception, une modélisation UML complète de la cokerie à été réalisée. Celle-ci comprenait un diagramme des cas d'utilisation, des diagrammes d'activités, de séquence, de collaboration, … et un diagramme de classe. L'étude était essentiellement basée sur la modélisation métier de la cokerie et donc axée processus de production.

Elaboration : itération 1 et itération 2. La phase d'élaboration comprend l'itération 1 soit la rétro ingénierie des bases de données relationnelles existantes et actuellement en utilisation chez Carsid et l'itération 2 soit une modélisation UML complète de la cokerie basée sur la gestion des exigences et donc plus proche des besoins réels des utilisateurs du futur système. Il convient de noter que cette deuxième phase est donc plus proche de l'utilisateur final du futur système et s'éloigne de la modélisation du processus de production.

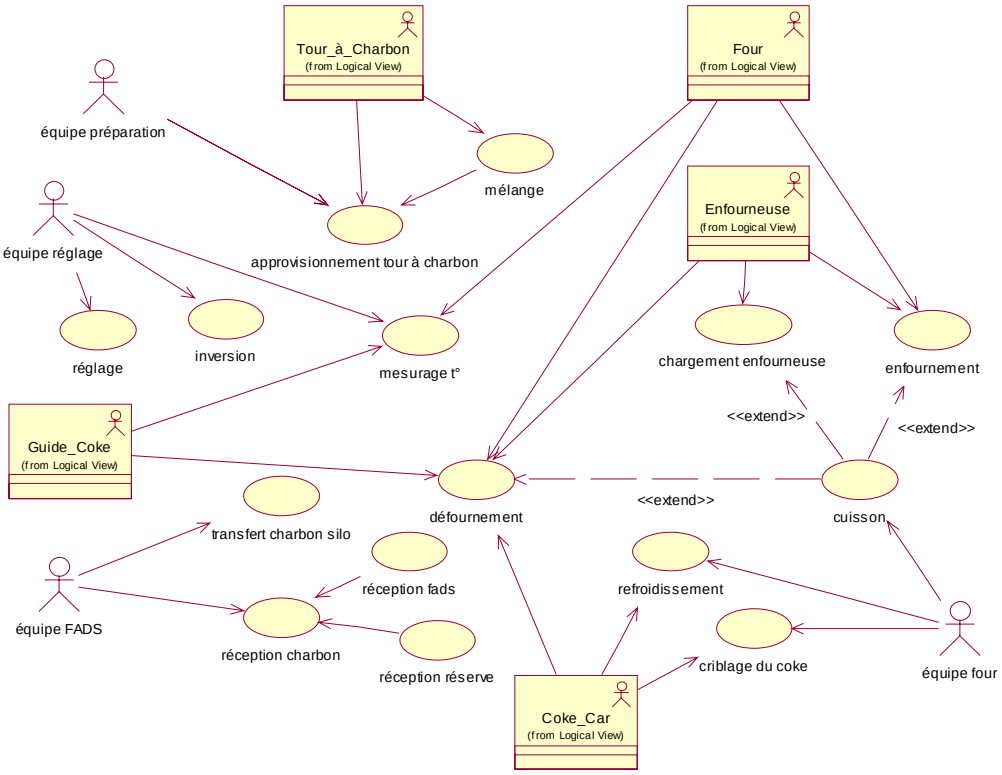

Figure 14 : Diagramme des cas d'utilisation de l'itération préliminaire.

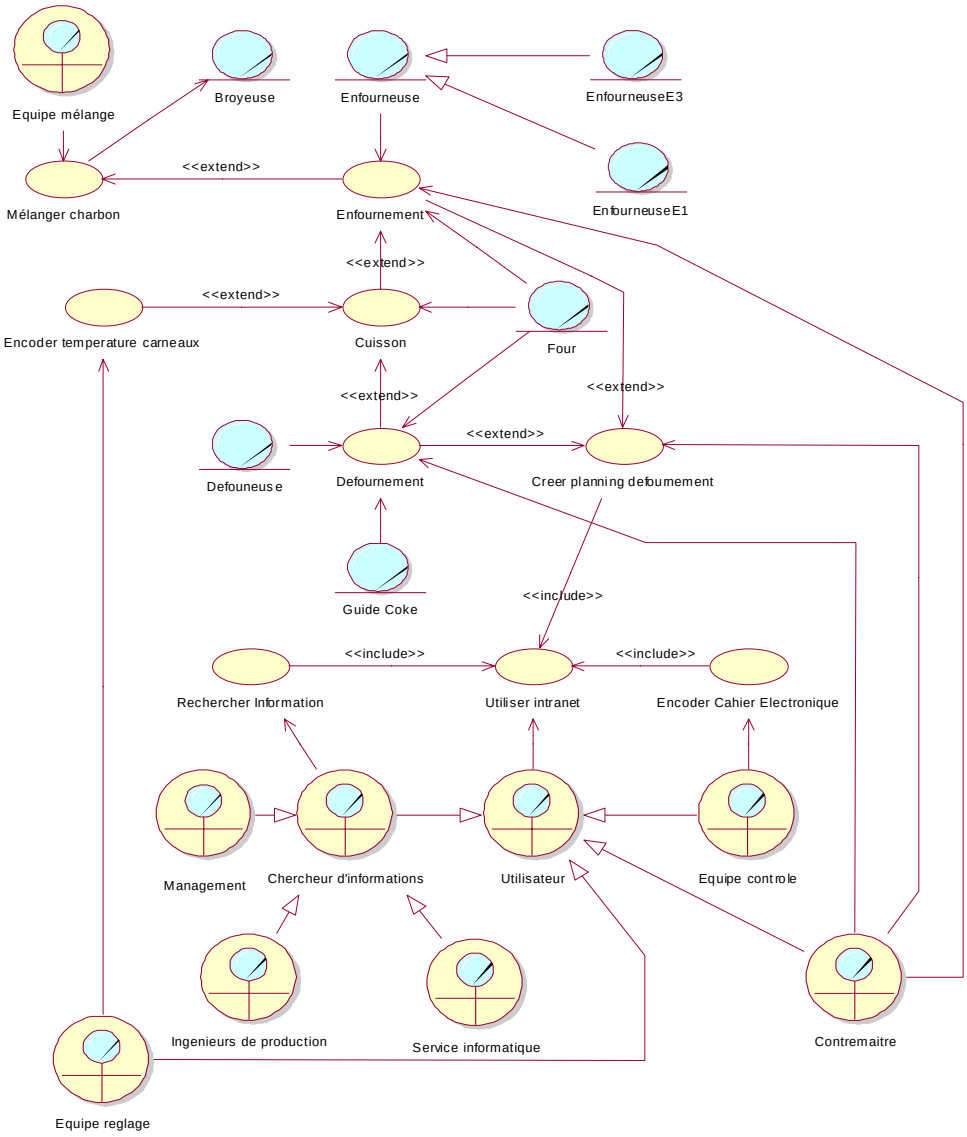

Figure 15 : Diagramme des cas d'utilisation de l'itération 2.

Les itérations : comparaisons et évolution du modèle.

Diagrammes des cas d'utilisation. Les diagrammes des cas d'utilisation ont été présentés dans les figures 14 et 15. Le lecteur peut constater que les cas d'utilisation repris dans le diagramme de l'itération préliminaire sont essentiellement axés sur les processus de production, tandis que ceux de l'itération 2 sont plus proches de l'utilisateur du système informatique.

Diagrammes d'activité. Un *workflow* est une séquence d'activités qui s'enchaînent au long d'un processus. Lors de l'itération préliminaire, le *workflow* de la cokerie de Carsid avait été représenté à l'aide de diagrammes d'activité qui permettent de représenter les enchaînements et les synchronisations nécessaires au bon fonctionnement du travail de production de coke (Figure 16).
On peut comparer cette séquence d'activité au *workflow* résultant de l'itération 2 (Figure 17). Ce

workflow est constitué de trois diagrammes d'activité complémentaires et successifs, liés aux trois cas d'utilisation *Mélanger charbons*, *Enfournement* et *Défournement.*
L'approche des deux *workflows* est assez différente, bien que certaines activités soient assez similaires. On notera, une nouvelle fois, que dans l'itération 1, l'accent est porté sur le processus de production, tandis que dans l'itération 2, il est porté sur les relevés d'information, les calculs, l'enregistrement et la validation de données.

Diagrammes de classe. Lors de l'itération préliminaire Aymeric Donnay et l'équipe ISYS ont générés un premier un diagramme de classe basé sur la modélisation métier de la cokerie. Ce diagramme de classe est présenté en figure 18.
A l'issue de la deuxième phase du projet Carsid, soit de l'itération 1, Marti Ibarz a réalisé par rétro ingénierie des bases de données existantes un nouveau diagramme de classe assez différent de celui produit lors de l'itération préliminaire, mais plus proche de la réalité des systèmes informatiques actuels de Carsid. Conçu au départ de bases de données relationnelles, ce diagramme ne présente des classes et des attributs mais pas de méthodes. Ce diagramme de classe est présenté en figure 19.
Enfin, sur base des modèles définis lors de la dernière phase, soit l'itération 2, un diagramme de classe plus complet a été élaboré. Ce dernier est présenté en figure 20 et 21. Ce diagramme s'architecture autour de la rétro ingénierie des bases de données de Carsid, mais présente également des méthodes répondant aux besoins de l'utilisation du système et définis lors de l'étude de gestion des exigences de la cokerie, de modélisation métier de la traction et de modélisation métier de la bascule.

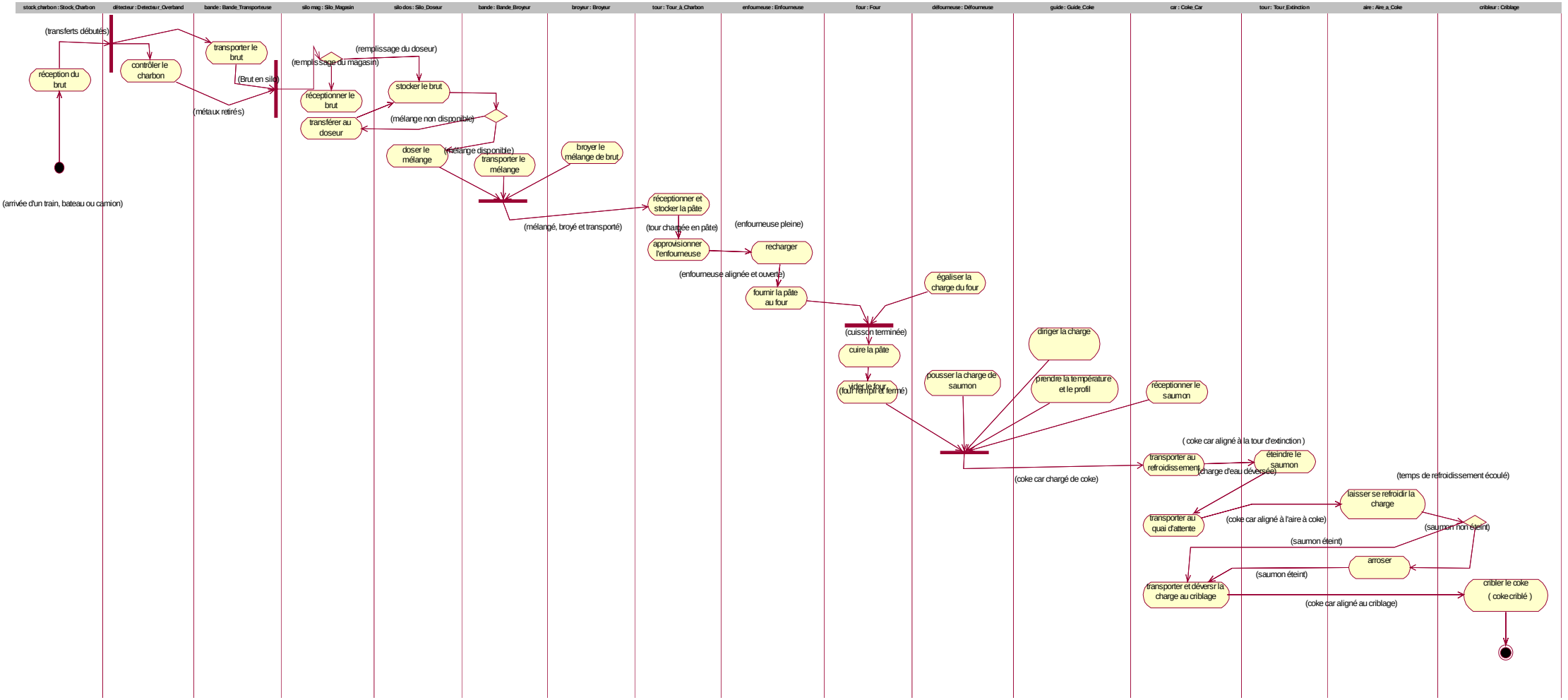

Figure 16 : Diagramme d’activités de l’itération préliminaire.

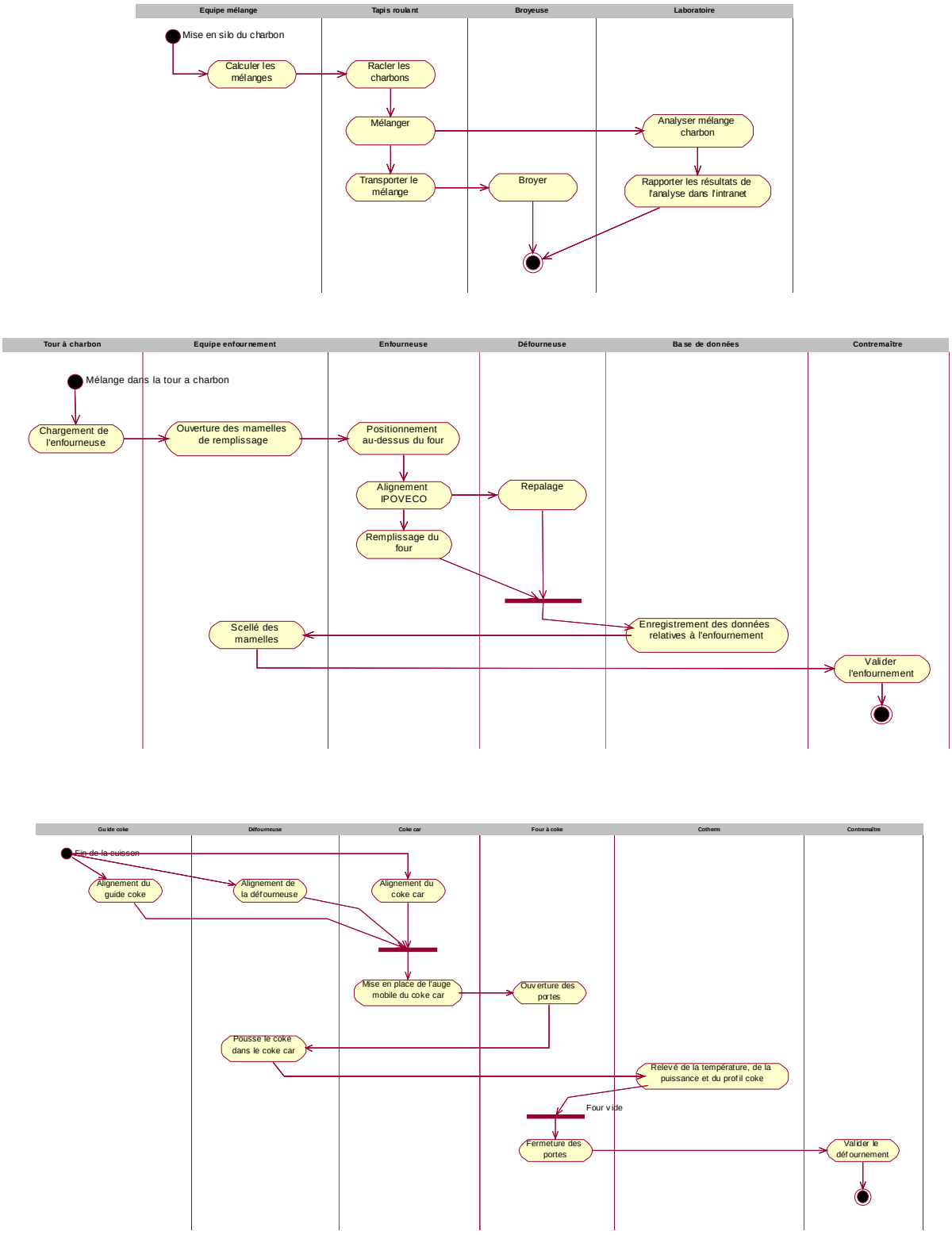

Figure 17 : Diagramme d'activités de l'itération 2.

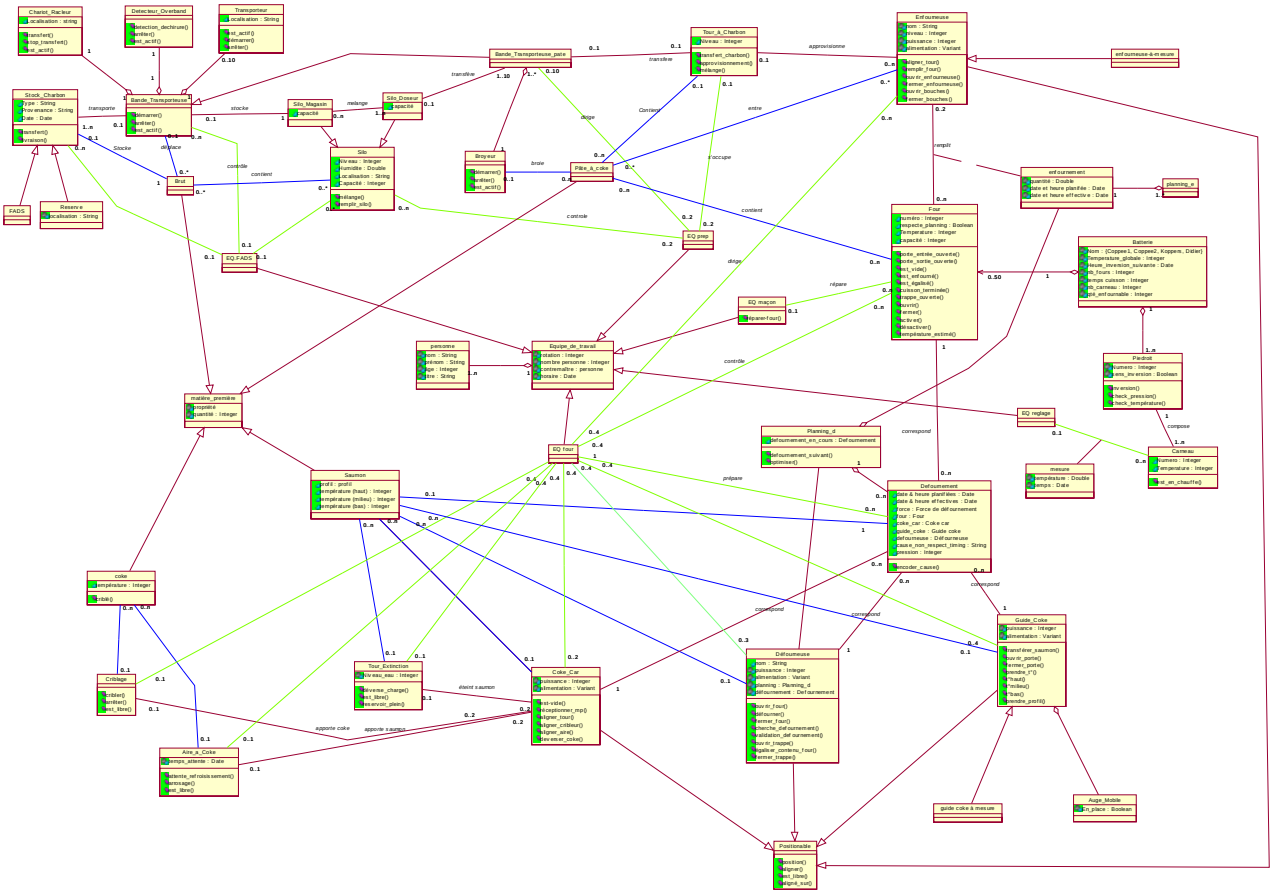

Figure 18 : Diagramme de classe de l'itération préliminaire.

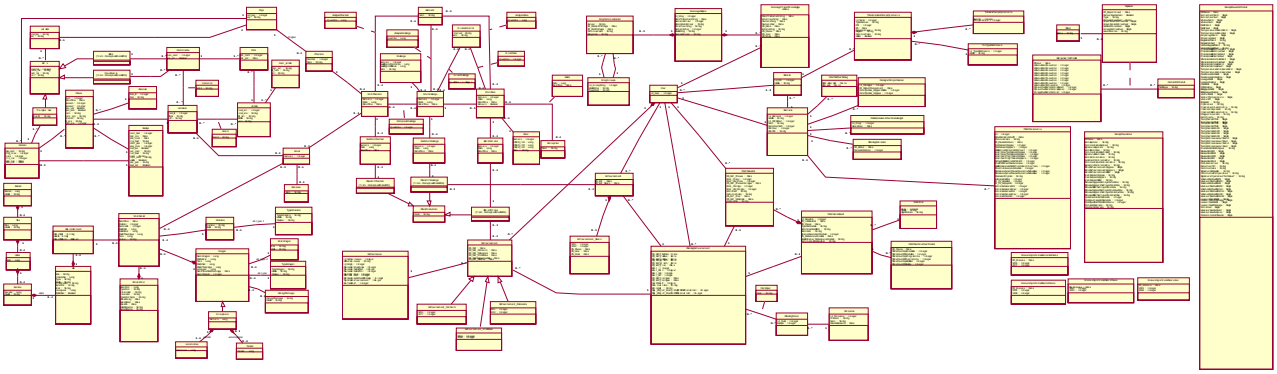

Figure 19 : Diagramme de classe de l'itération 1.

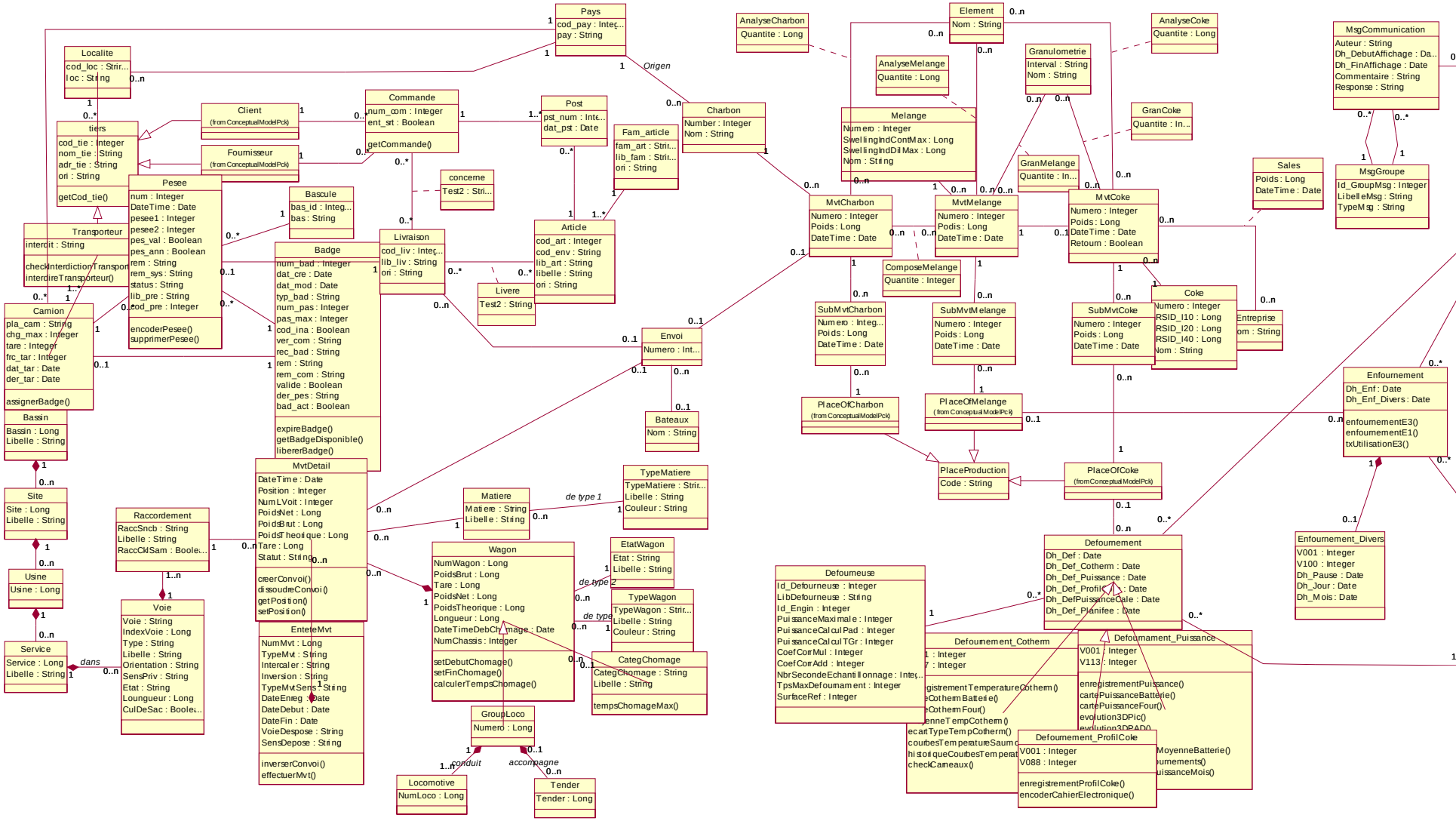

Figure 20 : Diagramme de classe de l'itération 2 – première partie.

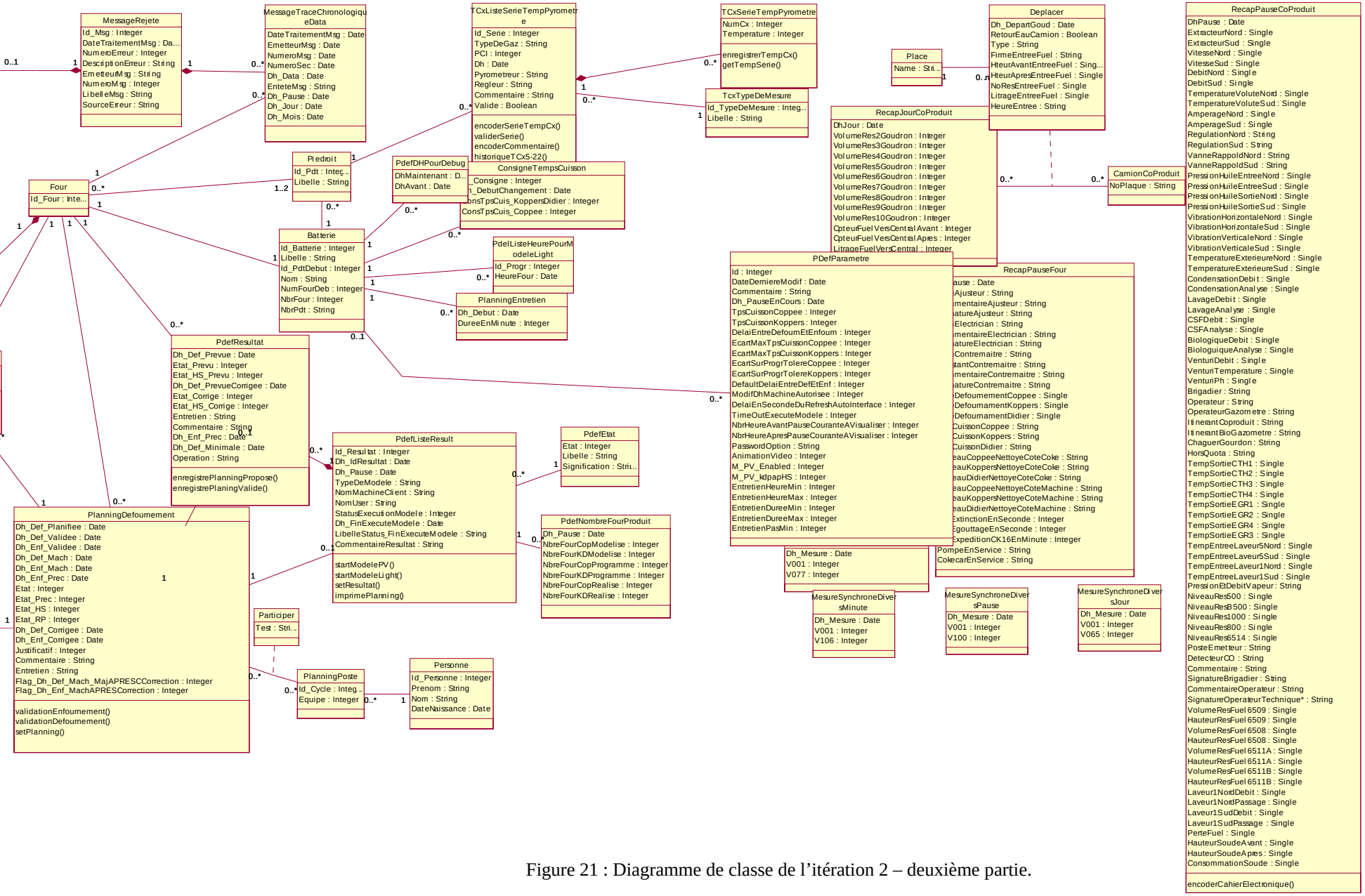

Figure 21 : Diagramme de classe de l'itération 2 – deuxième partie.

6 CONCLUSION

Les trois premières phases du projet de modernisation des systèmes informatiques de Carsid sont maintenant terminées. Chacune s'est déroulée dans un contexte différent mais elles font toutes parties d'un processus global et ont donc une cohérence propre.

Après avoir définit les concepts théoriques relatifs à la gestion de projet en général et au UP en particulier, cette méthodologie a été appliquée au projet de modernisation de la cokerie de l'entreprise Carsid. Les différentes activités et itérations menées dans ce cadre ont été exposées, ce qui a permis de mettre en lumière l'architecture bidimensionnelle du UP.

Les travaux de modernisation du système informatique sont toutefois loin d'être terminés. En effet, seules les activités de modélisation métier, de gestion des exigences, d'analyse et de conception ont été réalisées et il est maintenant raisonnable de penser à la phase d'implémentation. Aussi, le modèle est perfectible. Il est en effet possible d'approfondir les activités de modélisation notamment en travaillant plus amplement avec l'équipe informatique de Carsid pour obtenir une validation des modèles et de nouveaux détails a englober dans ceux-ci. Les travaux actuels n'ont également que peu abordé certains aspects tels que les co-produits et l'entretien où des travaux de modélisation pourraient être réalisés.

REFERENCES

Anonyme, *Synthèse des applications informatiques à la cokerie de Charleroi* (document interne), CARSID, Marchienne-au-Pont, 2002.

Anonyme, *La cokerie de Marchienne* (document interne), groupe Cockerill Sambre, Marchienne-au-Pont.

G.Booch, I.Jacobson, James Rumbaugh, *The unified modelling language user guide*, Addison Wesley Pub Co, 1998.

J. Caelen, *Rationaliser la conception participative*, www-geod.imag.fr/jcaelen/ transparents_fichiers%5CRUP-fr.ppt.

A. Donnay, M. Kolp, D. Massart, T. T. Do, S. Faulkner, and A. Pirotte. *Modélisation orientée objet de la cokerie de CARSID*. Rapport final Carsid, Août 2002.

A. Donnay, F. Fouss, M. Kolp, D. Massart, A. Pirotte, *Analyse orienté objet de processus sidérurgiques de type cokier*, Working Paper IAG 86/03, Université Catholique de Louvain, Mars 2003.

F. Fouss, M. Ibarz, M. Kolp, *Object-oriented reengineering of the steel production databases at Carsid*, Working Paper IAG 85/03, Université Catholique de Louvain, Mars 2003.

P. Kruchten, *The Rational Unified Process An Introduction. Second Edition*, Addison-Wesley, Upper Saddle River, NJ, Juin 2001.

L. Louvigny, *Modélisation orienté-objet d'aspects comportementaux de base de données. Application à la cokerie de Carsid*, Mémoire-projet IAG-UCL, Louvain-la-Neuve, 2003.

J. Rumbaugh, G. Booch, I. Jacobson, *The unified modelling language reference manual (Addison-Wesley object technology series)*, Addison Wesley Pub Co, 1998.

Y. Wautelet, *Application de la méthodologie RUP/UML à l'entreprise sidérurgique Carsid*, Mémoire-projet IAG-UCL, Louvain-la-Neuve, 2003.