STMicroelectronics - Ensimag

Rapport de Projet de Fin d'Études

Gaïa Projet : Application pour l'Optimisation de l'Overlay en Photolithographie

Antoine Duparay

12/8/2013

Coordonnées des acteurs

Entreprise

Raison sociale	STMicroelectronics
Adresse	850 rue Jean Monnet F-38 926 Crolles Cedex
Adresse du service où se trouve l'apprenti	Photolithographie – Crolles 2

Maître d'Apprentissage

Prénom et nom	Auguste LAM	
Tél.	04 38 92 21 33	
Adresse électronique	Auguste.lam@st.com	

Tuteur Pédagogique

Prénom et nom	Joelle Thollot	
Adresse	Ensimag BP 38 402 Saint Martin d'Hères Cedex	
Tél.	06 71 98 16 42	
Adresse électronique	Joelle.thollot@imag.fr	

Apprenti

Prénom et nom	Antoine DUPARAY	
Adresse	Chemin de Blardet 73 800 Les Marches	
Tél. personnel	06 65 11 10 19	04 38 92 32 17 (à ST)
Adresse électronique	Antoine.duparay@ensimag.fr	

Avant-propos

Le rapport suivant portant sur un domaine pointu, il utilise un vocabulaire spécifique afin d'obtenir la précision nécessaire au sujet. Par la suite, lorsqu'un mot qui semble devoir être redéfini sera rencontré pour la première fois, celui-ci sera écrit en italique. Cela signifie qu'une définition rigoureuse du terme se trouve dans le lexique à la fin du rapport. Veuillez vous reporter à celui-ci si son sens vous échappe.

Bien qu'étant évités au maximum dans ce document, certains anglicismes n'ont pas de traduction française ou celle-ci est rarement utilisée. Certains termes apparaissent donc en langue anglaise dans le texte. Ils sont généralement redéfinis dans le lexique à la fin du rapport.

Sommaire

Introduction

STMicroelectronics est une société franco-italienne productrice de composants semi-conducteurs. Elle crée des circuits imprimés pour différents secteurs : automobile, téléphone portable, télévision, cartes à puce… Les produits créés sont à la pointe de la technologie et nécessitent une précision record lors de leur fabrication. De nombreux procédés sont donc mis en œuvre pour la contrôler afin de minimiser le nombre et l'impact des défauts de fabrication. A Crolles 300, les machines sont en place dans une salle blanche où le taux d'humidité, la pression et surtout la quantité de particules dans l'air sont précisément contrôlés. Les *scanners* y impriment les composants sur des plaques appelées *wafers.* Ce processus est appelé *lithographie* (voir I.1.) et est effectué par couche successive. La mesure de l'erreur entre deux couches est appelé *overlay* (voir I.2). C'est cette erreur que nous cherchons à minimiser.

Les équipements présents en salle sont vendus par des sociétés externes qui fournissent un support humain, matériel et logiciel à ST. Le bon fonctionnement des machines est surveillé par les logiciels des équipementiers qui permettent de travailler sur les données enregistrées durant le processus de fabrication. Chaque logiciel étant spécifique à un équipement, ils ne sont pas capables de communiquer entre eux : les données ne sont pas dans des formats interopérables. En conséquence, il est impossible de travailler sur les données de plusieurs équipements dans un seul logiciel. De plus, leurs fonctionnalités sont spécifiques à leur équipement et ne répondent pas exactement aux besoins de ST. Le projet Gaïa a donc vu le jour avec comme objectif principal d'unifier en un seul logiciel toutes les données pouvant être enregistrées sur les différents équipements de la lithographie. Il vient aussi enrichir les actions possibles sur les données en collant au mieux aux besoins des utilisateurs puisqu'il évolue en suivant leurs remarques sur le modèle des méthodes Agile.

Depuis mon arrivée dans l'entreprise et jusqu'à la fin de mon alternance, je suis seul en charge de ce projet. Après avoir étudié la fabrication des composants et les besoins des utilisateurs afin d'avoir une bonne idée du fonctionnement envisagé de l'application, j'ai développé un prototype (version 0) qui a évolué au fur et à mesure de ses présentations aux utilisateurs et de l'ajout de fonctionnalités. Cette phase a duré durant mes deux premières années d'alternance, avec en moyenne une version majeure par période en entreprise (2 mois). Une fois que le prototype semblait convenir aux besoins et que les fonctionnalités importantes avaient bien été identifiées, mes sept mois de projet de fin d'étude ont été affectés à la création d'une version 1 de l'application avec un public cible plus large que les premiers utilisateurs. La première tâche dans ce cadre fut une refonte totale de la partie vue vers une interface plus séduisante, intuitive et peaufinée. La deuxième tâche a été de faciliter l'accès aux données. Dans le prototype, elles étaient exportées des logiciels fournis par les équipementiers et réimportées dans Gaïa à la main. Dans la version 1, elles sont maintenant importables en se connectant directement sur les bases de données des machines. Selon la stabilité de l'application à la fin de mon PFE, le code de celle-ci pourra être repris

directement ou servir de preuve de faisabilité pour permettre la prise en charge du projet par une véritable équipe IT.

Le présent document est développé en trois parties. La première présente toutes les notions nécessaires à la compréhension du projet : la photolithographie, l'overlay, la structure d'un wafer et le modèle scanner de correction des erreurs. La seconde partie décrit le projet lui-même, ses objectifs, comment il s'est déroulé, les outils et technologies utilisées, les difficultés rencontrées et il termine sur une petite partie technique en présentant les classes formant la partie modèle du logiciel. La dernière partie conclue en présentant l'application et ses fonctionnalités, d'abord en traitant de l'importation des données et de leur stockage, puis des possibilités existantes pour filtrer les données et ainsi cibler celles que l'on souhaite étudier en détail, et enfin de l'affichage de ces mesures ciblées dans une vue spécifique. Un dernier paragraphe traite d'une fonctionnalité supplémentaire : les lots virtuels.

Commençons donc par découvrir la notion clef du projet Gaïa : le fonctionnement de la photolithographie.

I Le contexte : La lithographie

I. 1. Le principe de la lithographie

Afin de bien cerner les objectifs du projet Gaïa, il est nécessaire de se pencher sur la manière dont fonctionnent les équipements de fabrication. Celui qui nous intéresse est appelée *cluster*. Un cluster contient deux sous machines indépendantes qui communiquent entre elles grâce à une interface : la piste (ou *track*) et le *scanner*. La piste est le point d'entrée et de sortie des lots, contrairement au scanner qui n'a pas de contact avec l'extérieur. La piste est donc la première à réceptionner un lot qui possède en général 25 plaques appelées *wafers*. Elle dépose sur les plaques, une par une, un anti-réflectif puis de la résine photosensible, c'est-à-dire qui réagit à une exposition à la lumière. La plaque passe ensuite dans un four afin de cuire la résine avant de rejoindre le scanner. Dans ce dernier, elle sera exposée champ par champ (voir le paragraphe structure d'un wafer) à une lumière ultraviolette à travers un masque. Ce dernier est constitué d'un motif dessiné sur du quartz. Si on compare à la photographique, le masque est l'équivalent du négatif : il contient l'image que l'on souhaite développer. La lumière ne passe qu'à travers certaines parties du masque et n'éclaire en conséquence la résine que sur certaines zones de la plaque. Une lentille est utilisée pour régler la netteté de l'image projetée sur la plaque. Elle permet aussi de rétrécir la taille de l'image par rapport à celle du masque. Ainsi, il est possible de dessiner sur la plaque un motif nettement plus petit que celui présent sur le masque.

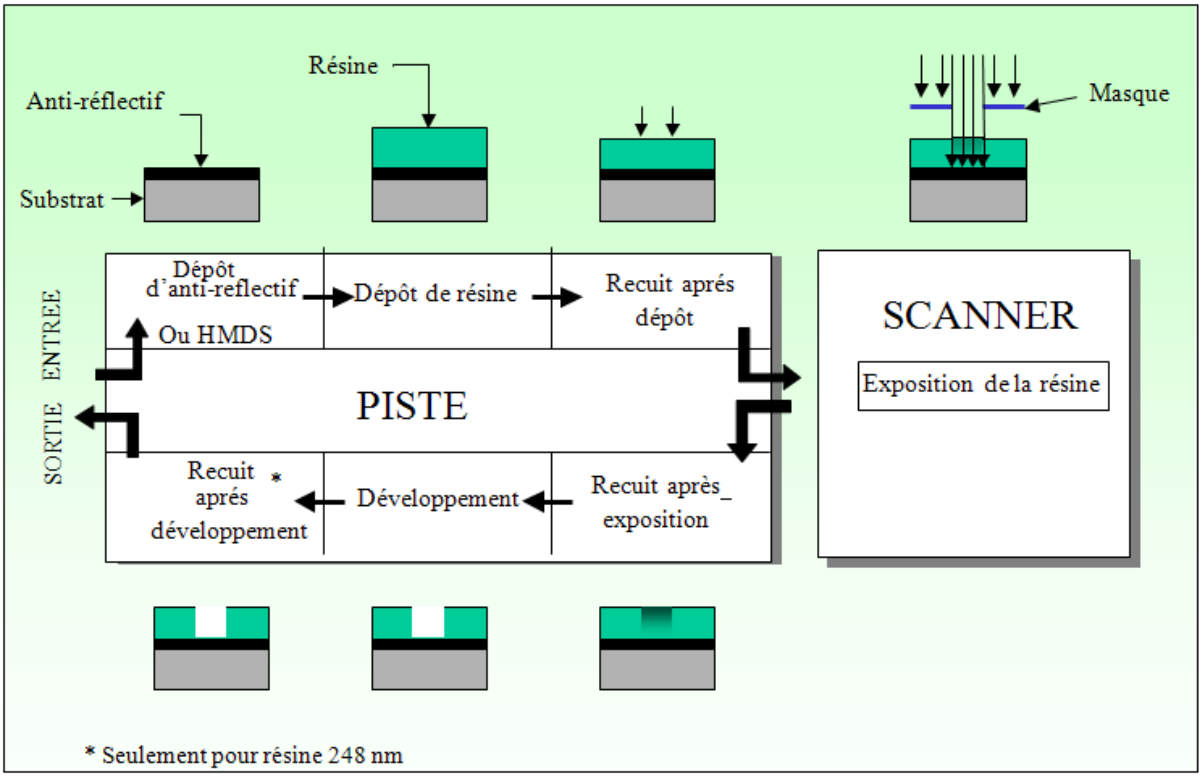

Figure 1 - Les différentes étapes du procédé de photolithographie

Après l'exposition, la plaque quitte le scanner puis effectue un nouveau passage dans un four. Vient ensuite la phase de développement, phase durant laquelle un substrat est utilisé pour supprimer, au choix, la résine qui a ou n'a pas été exposée. Ainsi, le silicium qui compose la plaque est maintenant protégé par la résine à certains endroits, et découvert à d'autres endroits. La plaque quittera ensuite le cluster pour aller à l'étape de la gravure, réalisée dans un autre équipement. La plaque ne subira la gravure qu'aux endroits qui ne sont plus protégées par la résine. Ces étapes sont réalisées de nombreuses fois, puisqu'un circuit imprimé est élaboré par couches successives. La plaque reviendra donc dans le cluster autant de fois que le circuit possède de couches.

I. 2. Les bases de l'overlay

Il est bien sûr nécessaire d'avoir une précision frôlant la perfection durant l'exposition de la plaque. En effet, si l'on observe un décalage trop important entre la zone que l'on souhaite éclairer et celle éclairée en réalité, ce décalage se répercutera entre la couche qui vient d'être exposée et la couche précédente. Il y a donc de fortes chances de voir apparaître un chevauchement des pistes qui conduisent le courant sur le circuit imprimé, ce qui provoquerait un court-circuit dans le produit final et le rendrait inutilisable. Des méthodes de contrôle rigoureuses ont été mises en place.

Il est donc nécessaire d'avoir le meilleur alignement possible entre les deux couches successives d'une plaque. En conséquence, le masque a été enrichi et ne possède pas seulement le motif du circuit que l'on souhaite créer sur cette couche mais aussi des *marques d'alignements* qui vont permettre au scanner d'exposer correctement la couche suivante. En effet, lorsque le scanner récupère une plaque, il commence par la positionner parfaitement par rapport aux circuits déjà dessinés. Pour ce faire, sa première action est de lire les marques d'alignement laissées sur la couche précédente à l'aide de lasers. La lecture de ces marques dîtes de références va permettre au scanner de placer la plaque exactement pour que l'exposition de la nouvelle couche soit alignée avec la couche inférieure. Les marques laissées sur la couche n – 1 permettent donc de l'aligner avec la couche n, qui elle-même possède des marques pour s'aligner avec la couche n + 1, etc. Ce mécanisme de réalignement entre chaque couche exposée minimise les erreurs d'alignement. L'*overlay* désigne la mesure du désalignement de chaque point du niveau exposé par rapport au niveau de référence. L'objectif étant bien sûr de minimiser cette erreur de désalignement pour éviter des dysfonctionnements dans les circuits gravés.

I. 3. Structure d'un wafer

Pour mieux comprendre les erreurs d'overlay possibles, nous devons d'abord nous intéresser plus précisément à la structure d'une plaque, ou wafer. Son diamètre est de 300mm, d'où le nom de l'usine : Crolles 300. Il est composé de différentes entités affichées dans le schéma ci-dessous. Dans l'application, chacune des entités correspond à une classe.

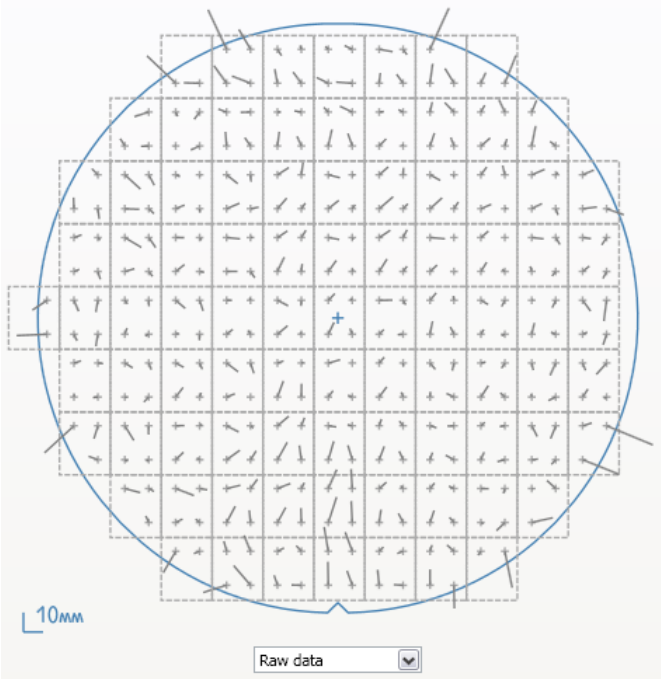

Figure 2 - Capture d'écran d'un plot de la vue vector plot view de Gaïa

Nous retrouvons ici un plot. Le wafer, reconnaissable à une encoche en bas de la figure, est dessiné en bleu. Une croix bleu indique son centre. Sa surface est quadrillée de manière à le découper en champs (ou *fields*) rectangulaires. Le champ contenant le centre du wafer est le champ d'indice (0 ; 0), celui à sa droite le champ (1 ; 0) et celui au-dessus de lui le champ (0 ; 1) : les champs constituent un repère orthonormé. Dans cet exemple, chaque champ contient quatre cibles, ou *target*, représentées par une croix grise. Ce sont les marques d'alignement présentées dans le paragraphe sur l'overlay. C'est l'écart entre l'emplacement prévu et l'emplacement réellement mesuré des cibles qui est enregistré. Cet écart est affiché sous forme de vecteurs qui indiquent la valeur de l'erreur et sa direction. La longueur du vecteur est directement proportionnelle à la valeur de l'erreur. Certains vecteurs sont particulièrement grands (ils représentent donc une grosse erreur) : cela est probablement dû à une erreur de mesure plutôt qu'à une véritable erreur d'alignement. Gaïa possède un système de filtre qui permet d'éliminer les erreurs de mesure pour n'afficher que les mesures valides à l'utilisateur.

I. 4. Le modèle scanner

Un *modèle* mathématique est constitué d'une série de paramètres calculés selon les mesures effectuées et qui permettent de déduire une correction à appliquer à la machine pour minimiser les erreurs et donc améliorer la qualité. Le modèle produit par le scanner possède dix paramètres qui correspondent chacun à un type d'erreur d'overlay. Ce n'est pas le seul modèle existant, mais c'est le plus utilisé.

Les dix paramètres du modèle scanner :

- Les six paramètres inter-fields :
  Ces paramètres sont à l'échelle du wafer et donc communs à tous les champs.
  - **Translation** (en X et en Y) : La translation a lieu lorsque la plaque est droite mais décalée par rapport au référentiel.
  - **Scale** (en X et en Y) : L'image imprimée est trop grande ou trop petite par rapport à celle souhaitée.
  - **Non-orthogonalité wafer** : Une rotation en X et en Y mais avec un angle de signe différent.
  - **Rotation wafer** : La rotation survient lorsque l'image est bien centrée mais avec un angle par rapport à celle voulue.
- Les quatre paramètres intra-fields :
  Ces paramètres sont à l'échelle du champ.
  - **Magnitude** (grandissement) : Le même en x et en y
  - **Asymag** (grandissement asymétrique) : x et y ont la même valeur mais un signe opposé
  - **Rotation field** : Rotation au niveau du champ
  - **Orthogonalité field** (rotation asymétrique) : Rotation de signe différent entre x et y

Le modèle scanner permet de déduire des données mesurées une valeur pour chacun de ces paramètres. Ces valeurs correspondent à la correction optimale qu'il est possible d'appliquer à la machine en vue de réduire les décalages entre l'image exposée et le référentiel.

Lorsqu'on calcule un modèle, on a donc trois types de mesures disponibles :

- les données brutes, celles qui ont été enregistrées par la machine
- les corrections optimales calculées par le modèle
- le résiduel, c'est-à-dire la soustraction des données brutes par la correction calculée, ce qui correspond au résultat espéré pour la prochaine mesure effectuée sur la machine.

Pour chaque modèle différent, nous avons donc à partir des mêmes données brutes une correction et un résiduel différent.

Le modèle scanner se calcule sur un échantillon de mesures correspondant au wafer. Il permet d'obtenir ainsi la meilleure correction possible pour le wafer. Cependant, cette correction est théorique. En effet, pour l'appliquer, il convient de modifier les réglages de la machine. Or, l'équipement fonctionne lot par lot : cette manipulation n'est pas réalisable entre chaque plaque mais qu'entre chaque lot. Le modèle peut donc être calculé de deux manières différentes : par

wafer, c'est-à-dire que le modèle est calculé pour chaque wafer et appliqué à chaque wafer, c'est la meilleure correction mais elle reste théorique, et par lot, c'est-à-dire que le modèle est calculé pour chaque wafer puis, à partir de chacune de ces corrections, la moyenne est effectuée pour calculer la meilleure correction globale au niveau du lot. Les valeurs de la correction sont alors les mêmes pour chaque wafer d'un même lot, c'est ce qui se passe en réalité.

De plus, sur chaque cible est mesurée l'erreur en X et en Y. Lorsqu'on calcule un modèle, on a donc en réalité 10 valeurs disponibles :

- La mesure originale (celle importée, brute), en X et en Y
- La correction calculée pour cette mesure pour le wafer, en X et en Y
- Le résiduel correspondant à cette correction, en X et en Y
- La correction calculée pour cette mesure pour le lot, en X et en Y
- Le résiduel correspondant à cette correction, en X et en Y

La correction du wafer étant optimal mais théorique, celle du lot est une moyenne de celles des wafers, moins bonne mais applicable.

Nous avons à présent fait un tour rapide des notions qui entourent le projet Gaïa. L'objectif affiché est de fournir aux ingénieurs de photolithographie les outils nécessaires à la surveillance de l'overlay durant la fabrication des plaques.

II Le projet Gaïa

II. 1. Le contexte du projet

Créer un outil informatique puissant a un coût. Il est important pour l'entreprise que le logiciel soit rentable, c'est-à-dire que l'augmentation de l'efficacité des utilisateurs apportée par l'application soit supérieure au coût de la mise en place de l'outil. Lorsque l'on perçoit que l'environnement de travail pourrait être amélioré, il n'est pas évident d'évaluer l'apport en performance d'une éventuelle meilleure solution : il apparaît clair que les outils en place ne sont pas optimaux, mais la vision d'un autre outil est souvent vague, et il n'est pas possible techniquement de chiffrer le gain qu'il apporterait. Dans ces conditions, il n'est pas facile de débloquer un budget souvent important pour développer une meilleure solution.

Une approche possible à ce problème est de créer un prototype pour tester les idées d'amélioration que pourrait apporter la nouvelle solution. Moins contraint, moins robuste, sans les exigences de fiabilité que l'on peut attendre d'un projet traditionnel, un prototype est en conséquence nettement moins coûteux à développer. Il sert ainsi de preuve de faisabilité : les utilisateurs potentiels le découvre, émettent des suggestions, et il devient plus facile d'évaluer le nouvel outil, quelles seront ses caractéristiques, sa pertinence et le gain en efficacité qu'il pourrait apporter. Il devient possible en se basant sur les données récoltées pendant cette première étape de préparer une solution plus robuste et mieux adaptée. Le prototype a donc une importance vitale pour le projet : c'est sur lui que le futur cahier des charges sera basé et c'est sur lui que repose le dossier pour construire la solution pérenne. Et cela, tant du point de vue de l'apport du projet pour les utilisateurs (travail plus efficace et plus rapide mais aussi meilleurs résultats) que de celui des caractéristiques de la solution envisagée (fonctionnalités, technologies utilisées, performance…). Comme pour mon stage d'IUT au CERN à Genève, c'est cette approche qui a été choisie pour le projet Gaïa, et mon alternance de trois ans a été complètement dédiée à la création de ce prototype, de l'analyse du besoin au déploiement.

Ce contexte est très intéressant comme première expérience en entreprise : il permet d'être en contact direct avec les utilisateurs, ce qui est rarement le cas en tant que développeur dans une grande organisation (ce rôle est en général réservé à l'agent commercial ou au chef de projet). Il permet aussi de découvrir toute la chaîne par laquelle passe le projet, de l'analyse des besoins au déploiement en passant par la rédaction des spécifications, la conception, le développement et les tests. Cela oblige par la même occasion à savoir s'organiser et être rigoureux car travailler seul implique une grande autonomie. Tout n'est malheureusement pas positif, car il n'est pas possible d'avoir dans ce contexte des retours sur les décisions et les choix pris dans le développement du logiciel. De même, cela implique de travailler sans budget, ce qui peut être bloquant. Les problèmes auxquels j'ai dû faire face sont décrits plus en détail dans le paragraphe « les difficultés rencontrées ».

II. 2. La gestion du projet

II. 2. 1. Cycle de développement

Durant les deux premières années d'apprentissage, le projet s'est déroulé de manière assez classique. La première période en entreprise a été orientée sur analyse de l'existant, du contexte et du besoin et rédaction du cahier des charges. La seconde a permis le développement du cœur de l'application, du premier module d'importation puis de l'interface graphique. Les données ont été modélisées de manière suffisamment générique (voir Figure 2 diagramme de classe du modèle ci-dessous) pour permettre la représentation des différents types de données arrivant des différentes machines (il n'a pas été nécessaire de revoir le modèle au fur et à mesure que de nouveaux formats ont été inclus). A partir de cette base robuste, il a ensuite été décidé d'avancer sur le projet avec un cycle incrémental, qui se marie très bien avec le fonctionnement en alternance sur des périodes de deux mois : une période était tout simplement un nouveau cycle, une nouvelle itération dans l'avancement du projet. Ainsi, pour chaque période en entreprise, il y a eu succession des mêmes étapes : d'abord, la correction des bogues remontés par les utilisateurs pendant la période d'école, puis le choix d'une ou des nouvelles fonctionnalités avec l'analyse du besoin, suivi de la rédaction des spécifications puis de la conception du module et enfin du déploiement aux utilisateurs. La proximité des utilisateurs s'est avérée très stimulante et vraiment efficace pour la conception de l'application, que ce soit pour les fonctionnalités nécessaires et l'ordre dans lequel elles devaient être développées ou pour l'interface et les options / réglages par défaut importants pour les utilisateurs. Le rythme de déploiement était donc environ d'une version majeure par période de deux mois et de deux ou trois versions mineurs selon les bogues ou petites améliorations mis en évidence par les utilisateurs.

Pour le Projet de Fin d'Étude et cette dernière période de 7 mois, la manière de travailler a un peu évolué. En effet, nous avons essayé avec mon Maître d'Apprentissage d'impliquer un nombre plus large de personnes dans le projet. Le but était de concevoir le logiciel de manière à ce qu'il convienne à différentes utilisations. En effet, de nombreuses personnes travaillent sur la lithographie, mais pas au même niveau. À ST, il y a trois équipes différentes : Litho R&D team, Litho Process team et Litho Equipment team. Il est important d'inclure des personnes de chacune de ces équipes dans la boucle de développement dès le début car leur cas d'utilisation sont passablement différents : alors que l'équipe R&D cherche à améliorer la gestion de l'overlay sur le long terme en essayant de trouver de nouvelles pratiques, l'équipe Equipment est responsable de la maintenance de la machine et souhaite savoir à l'instant précis quel est le meilleur réglage possible pour minimiser l'erreur d'overlay. Gaïa doit donc être conçue de manière à permettre des usages assez différents. Ainsi, pendant le Projet de Fin d'Étude, le développement évolue toujours de manière itérative mais les retours ne se font plus uniquement en direct ou par e-mail : des réunions sont organisées régulièrement à mon initiative pour convier des personnes des différentes équipes et réfléchir ensemble aux points importants et à une approche correspondant à tous. Ces réunions permettent donc de montrer les avancées du projet et de fixer les priorités pour le développement d'ici à la prochaine réunion. Les utilisateurs ayant un rôle de clients, cette approche de gestion du projet a des points communs avec l'approche des méthodes agiles comme Scrum, où le client a un rôle important dans le cycle de développement du projet.

II. 2. 2. Rétro planning

Mon projet de fin d'étude a commencé le 11 février 2013 et s'est terminé le 6 septembre 2013. Étant seul à travailler sur le projet, mon emploi du temps était assez flexible, ce qui m'a permis de m'adapter à différents contretemps (voir le paragraphe « Difficultés rencontrées »). Le travail sur le projet s'est donc déroulé de la manière suivante pour les sept mois du PFE :

- Fin février : Correction des problèmes repérés par les utilisateurs sur la version 0 de Gaïa pendant la période d'école du début de l'année.
- Mars – Avril : Création de la nouvelle version de Gaïa : **nouvelle structure de l'application** en place. **Réécriture des contrôleurs et des vues en utilisant le Ribbon** fourni par Microsoft pour avoir une interface similaire à Microsoft Office. **Importation des fonctionnalités de base de Gaïa** depuis la version 0 (importation des données depuis des fichiers XML, calcul des statistiques et du modèle Scanner, affichage des données sur le plot…). En parallèle, rédaction du pré-rapport pour l'école.
- Mai – Juin : **Intégration de l'API KT-Analyzer** fournie par KLA. Plusieurs échanges avec le support ont été nécessaires, en parallèle, le développement continue : d'autres bogues ont été corrigés sur la version 0 et une nouvelle fonctionnalité permettant **d'exporter les données en CSV** a été développée (ajoutée à la version 0 puis réimportée dans la version 1). Mise en place de la **page d'accueil modélisant l'intégralité des données** importées dans l'application dans deux graphiques. Ces derniers ont été dessinés grâce à une bibliothèque de modélisation financière qui a été incluse dans Gaïa. **Importation de la vue des données brutes** depuis la version 0. Visite de la tutrice pédagogique.
- Juillet : Mise en place du module de communication avec la base de données « Dataware », une copie de la base de données de la salle blanche. L'application est à présent capable de récupérer le contexte dans lequel les lots sont passés sur les machines (information d'exposition, sur quelle machine, etc). Il devient possible de filtrer les données affichées sur le graphique principal en se basant sur ces critères.
- Août : Importation depuis la version 0 de la notion de **lots virtuels** (lots inexistants en réalité créés dans l'application à des fins de simulation), **peaufinage** de l'application et correction de bogues, début de rédaction d'un manuel utilisateur et préparation du **rapport et de la soutenance** pour l'école.
- Début septembre : **Nettoyage et ajout de commentaire** au code pour un futur mainteneur, fin de rédaction du **manuel utilisateur**.

En parallèle, l'habituel travail de correction des bogues et ajout de petites suggestions a continué, tant sur la version 1 que sur la version 0, toujours utilisée tant que la version 1 n'était pas assez complète.

II. 3. Les outils utilisés

II. 3. 1. Le besoin et le choix de la technologie

Un certain nombre de besoins ont été mis en évidence par la phase d'analyse. Tout d'abord, les données vont venir de différents équipements, sous différentes formats. Il faut donc une technologie flexible afin de pouvoir s'adapter à chaque type de source sans avoir à repartir de zéro, que les données proviennent de fichiers XML, de CSV, d'une base de données… Cette souplesse permet la réutilisation de code entre chaque module d'importation. Ensuite, le volume de données à traiter étant important, il est nécessaire d'avoir une technologie performante, afin d'offrir un service toujours fluide à l'utilisateur malgré la quantité d'informations. De plus, l'application sert entre autre à faire de la visualisation de données, notamment à travers des représentations graphiques. Comme le jeu de données a une précision importante, il faut que l'affichage graphique soit correct et permette une observation détaillée. Une plateforme graphique robuste est donc nécessaire, avec un affichage vectoriel capable de dessiner simplement toutes les formes possibles. Mon Maître d'Apprentissage a donc choisi la technologie .Net (C#), un langage objet donc flexible et assez performant. C# est relativement comparable à Java. De plus, .NET est beaucoup utilisé par le département informatique d'ST : la technologie est donc assez standard dans l'entreprise. L'écosystème est très complet et contient des solutions qui correspondent bien au besoin : la technologie WPF (Windows Presentation Foundation) permet de construire des interfaces graphiques flexibles les décrivant grâce à la syntaxe XAML (eXtensible Application Markup Language) et Linq est une solution robuste pour l'importation des données depuis des sources variées (Linq to XML, Linq to SQL, Linq to CSV…).

II. 3. 2. L'environnement de développement

Il n'y avait pas de budget disponible pour ce projet, tous les outils et technologies utilisées devaient donc être gratuits et autorisés pour un usage en entreprise. Au niveau de l'éditeur de code, le choix s'est porté tout naturellement sur Visual Studio en version Express, outil attitré pour développer en C#. Les diagrammes UML modélisant l'application ont été réalisés avec StarUML, logiciel sous licence GPL. Le diagramme de classe du modèle est consultable ci-dessous. Pour permettre un suivi de l'évolution du code et de sa complexité, j'ai utilisé l'outil Libre SourceMonitor. Cet outil parcourt le code et fourni des statistiques permettant de cibler les parties méritant d'être retravaillées. Il permet par exemple de connaître le nombre de fichiers, de classes, de méthodes, de lignes de code, mais aussi le pourcentage de commentaires et de documentation, le nombre moyen et maximum de méthodes par classe et la complexité moyenne et maximum des classes et des méthodes. Par exemple, le projet Gaïa est aujourd'hui composé de 97 fichiers pour au total 9636 lignes avec une moyenne de 7 ,55 méthodes par classe. SourceMonitor permet aussi de naviguer facilement entre les méthodes pour découvrir celles qui sont trop longues ou trop complexes. Avant la publication de chaque nouvelle version, j'utilisais donc cet outil pour garder au maximum un code clair et maintenable. J'ai de plus implémenté des tests unitaires pour les parties « critiques » de l'application : le modèle et la base de données locale. Ils permettent principalement de tester la non-régression de l'application lors de l'ajout de nouvelles fonctionnalités ou des remaniements du code de grande envergure.

II. 4. Difficultés rencontrées

Plusieurs difficultés sont apparues pendant le développement de ce projet. La première et certainement la plus grosse difficulté est due au contexte même du projet. Comme expliqué dans le paragraphe sur le contexte, le but de ces trois années d'alternance a été de développer un prototype d'application afin de juger de la viabilité du projet. En conséquence, j'étais seul développeur et intégré dans une équipe travaillant sur la lithographie, sans compétence informatique. Si ce contexte m'a appris l'autonomie et m'a permis d'avoir une vision globale du projet en intervenant à tous les niveaux, il a aussi été difficile de progresser sans aide : la seule ressource à ma disposition était un accès internet. J'ai pu de temps en temps contacter des personnes du service informatique qui m'ont beaucoup aidé, mais avaient peu de temps à m'accorder. Or, plutôt orienté Logiciel Libre et développement web, je ne connaissais pas du tout les technologies Microsoft en arrivant à STMicroelectronics. Je me suis donc retrouvé de nombreuses fois bloqué pendant plusieurs jours sur des problèmes qui auraient pu être corrigés rapidement si j'avais eu du support. De plus, j'ai essayé d'organiser au mieux mon code, mais personne n'a validé sa structure, qui aurait certainement pu être améliorée par quelqu'un ayant l'expérience de cette technologie. En plus du manque de compétence technique, l'absence de budget a aussi impliqué plusieurs complications, par exemple, il n'a pas été possible d'installer un environnement de test recréant la configuration des utilisateurs. Cette dernière n'étant pas la même partout (Windows XP ou Seven, .NET 3.5 ou .NET 4…), il a fallu tester directement sur les machines des utilisateurs le fonctionnement correct de l'application. Voici un exemple plus détaillé dans lequel l'absence d'un environnement de test a grandement retardé le déploiement de l'application.

Il faut d'abord savoir que comme Gaïa est pour l'instant un prototype, il n'est pas officiellement supporté par ST donc il n'y a pas de mise à disposition du logiciel par le service informatique, et donc pas de procédure officielle pour l'installer. Il faut que chaque utilisateur qui souhaite avoir Gaïa à disposition l'installe manuellement. Or, les utilisateurs ne sont pas administrateurs de leur machine. Une des contraintes du prototype était donc de pouvoir s'installer sans nécessiter les permissions administrateurs. Cette contrainte avait été identifiée dès la phase d'analyse et donc prise en compte dans le processus de développement.

Gaïa est capable de communiquer directement avec l'équipement de QArcher de KLA Tencor grâce à une API fournie par l'entreprise. Cette API a été installée sur les postes de travail de mon Maître d'Apprentissage et de moi-même, à des fins de développement. Gaïa empaquette donc l'API dans une librairie (DLL) contenant les méthodes nécessaires pour parler à l'équipement. J'ai testé cette configuration avec succès sur mon ordinateur, puis mon Maître d'Apprentissage a testé à son tour Gaïa avec l'API embarquée et a confirmé que la communication avec la machine s'effectuait correctement. Le développement s'est donc déroulé sur cette base et a correctement progressé, le support de KLA étant assez réactif lorsque je rencontrais des points bloquants sur l'utilisation de l'API. Après plusieurs semaines de développement, la fonctionnalité marchait correctement, et j'ai finalement mis la nouvelle version de l'application à disposition des utilisateurs. Et là, surprise, impossible pour Gaïa de communiquer avec l'équipement. L'application s'installe correctement et ne plante pas, mais affiche le message prévu en cas de connexion impossible (comme si l'ordinateur n'était pas relié au réseau). Après étude, il s'avère que l'installation de l'API de KLA avait ajouté des clefs au registre Windows, clefs contenant des informations comme les identifiants de connexion à la machine. L'API n'est donc pas capable de fonctionner sans ces clefs de registre supplémentaires,

et pour modifier le registre, il faut bien sûr être administrateur de sa machine. En conséquence, sans être administrateur, ou du moins, sans installer manuellement l'API avant d'installer Gaïa, l'application perd une de ces fonctionnalités principales, de laquelle dépendent toutes les autres : la facilité d'accès aux données sur lesquelles Gaïa permet de travailler (le seul moyen restant pour importer les données étant les fichiers XML à importer un par un manuellement, processus que nous cherchions justement à automatiser).

Cet exemple illustre bien comme il est important d'avoir un bon environnement de test : tester la communication depuis mon poste de travail puis celui de mon maître d'apprentissage n'était pas suffisant puisque nous avions tous les deux l'API d'installée, ce qui n'était pas le cas des utilisateurs. Une machine virtuelle vierge aurait permis de mettre en lumière le problème beaucoup plus tôt et d'ainsi adapter le développement en fonction. Finalement, la solution la plus simple est de demander à chaque utilisateur de se faire installer l'API, l'application est en effet officielle donc disponible même pour les personnes non administrateur. Cette solution est cependant beaucoup plus pénible pour l'utilisateur que la simple installation en quelques clics de Gaïa…

II. 5. Extrait d'architecture de l'application

L'application est organisée à partir du patron de conception MVC, pour Modèle Vue Contrôleur. Le modèle représente la modélisation des données du monde réel dans l'application. C'est aussi dans le modèle qu'on retrouve le code correspondant à la persistance des données. La vue est la partie affichée à l'utilisateur, et le contrôleur s'occupe des interactions entre les deux. Ces deux parties ont été entièrement réécrites dans Gaïa version 1, mais le modèle est strictement le même que celui de Gaïa version 0.

Le diagramme de classes ci-dessous représente la modélisation des données à l'intérieur de l'application. Comme nous pouvons le voir, nous retrouvons grâce au modèle objet les relations existantes dans la réalité. Un lot est composé de champs (les fields), de cibles (targets) et bien sûr de plaques (wafers). Pour un lot donné, tous les champs possèdent la même taille. De même, chaque champ aura les mêmes cibles (targets), ces données sont donc regroupées au niveau du lot, avec d'autres informations, comme les coordonnées du champ (0 ; 0) dans le repère du wafer. Les wafers ont un numéro de slot et des mesures associées, mesures qui ont une valeur (en complément d'autres informations) et sont rattachées à une cible et à un champ. Le lot est aussi rattaché à un fichier LER, pour Lot Exposure Report, qui contient les informations sur la manière dont le lot a été exposé. Le LER contient notamment un dictionnaire de « FieldExpoInfo », chacun d'eux correspondant à un champ et contenant ses informations d'exposition. Cette modélisation est assez générique pour permettre la représentation des données quel que soit l'équipement de provenance.

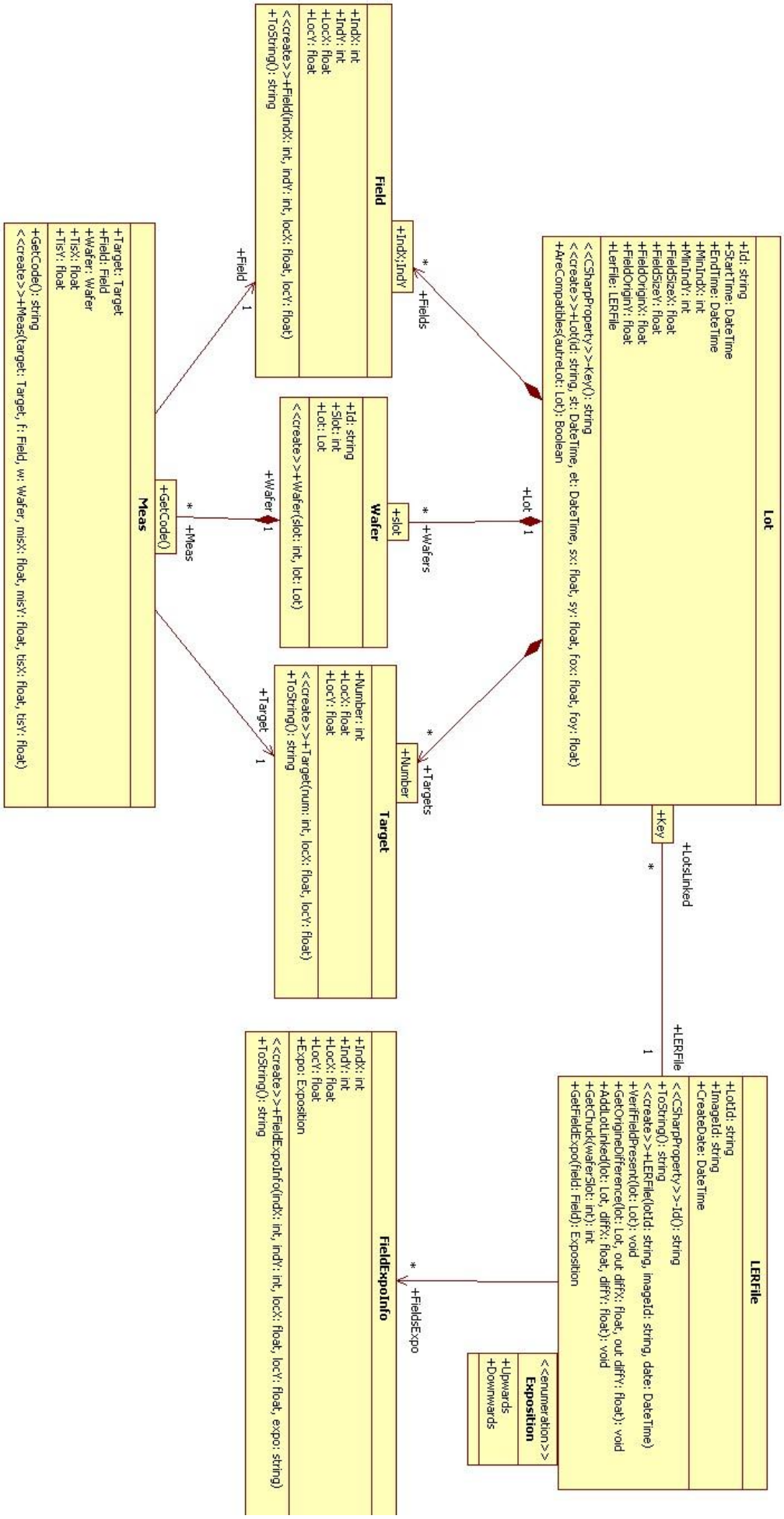

Figure 3 - Diagramme de classe du Modèle

III. À la découverte de l'application

Cette partie du rapport est consacrée à la description de l'application en elle-même. Elle présente à partir de captures d'écran les différentes fonctionnalités de Gaïa et compare les évolutions entre la version 0 créée pendant les deux premières années d'alternance et la version 1 développée pendant le Projet de Fin d'Étude.

III. 1. Vue d'ensemble

Commençons par observer une vue d'ensemble de la page d'accueil de Gaïa 0.42 (dernière version de la branche 0) et de Gaïa 1.1 (dernière version de la branche 1). Nous pouvons voir que même si l'on retrouve les mêmes éléments, ils sont présentés de manière complètement différente et les fonctionnalités offertes ont beaucoup évoluées.

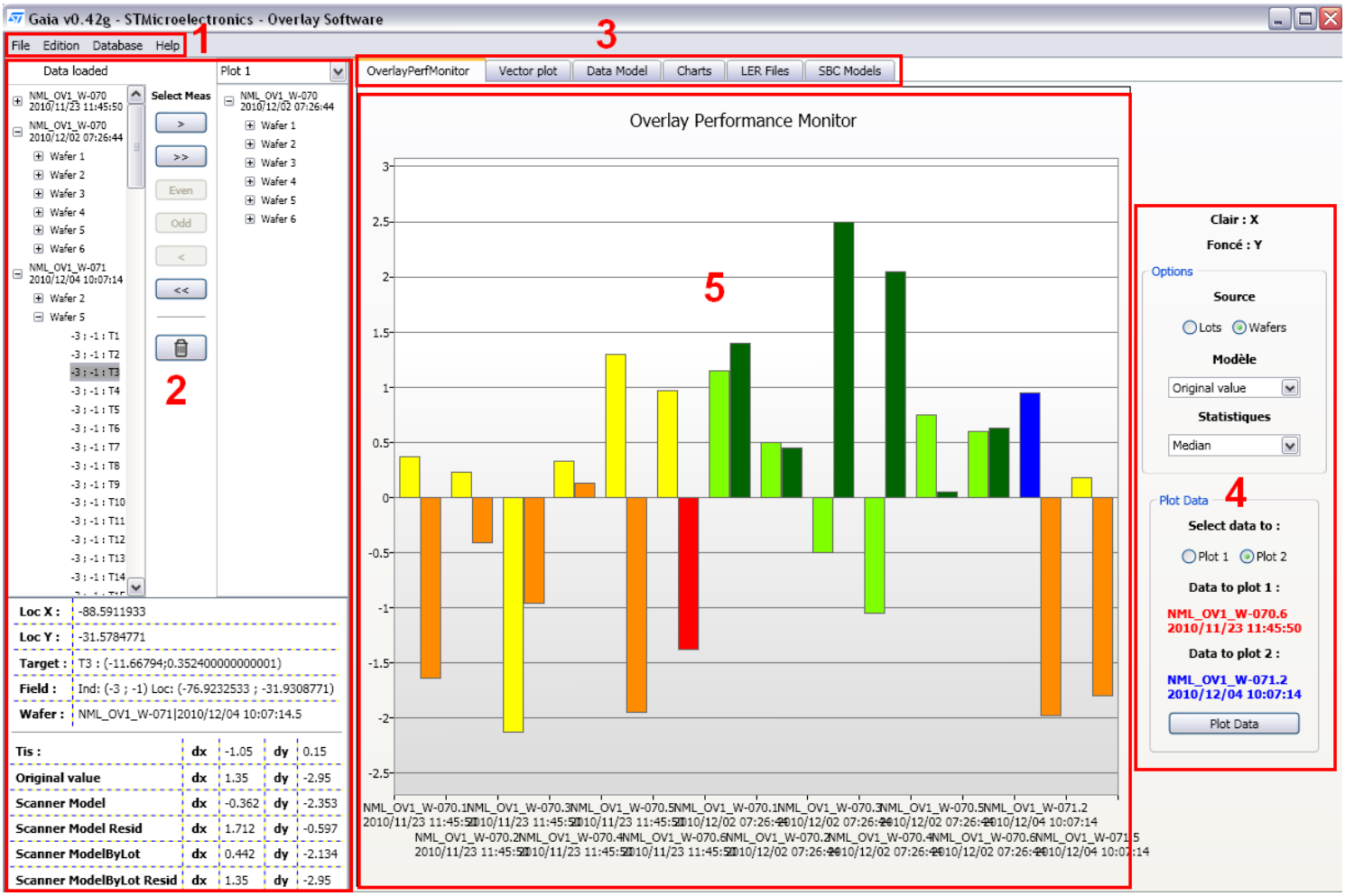

Figure 4 – Vue d'ensemble de Gaïa 0.42

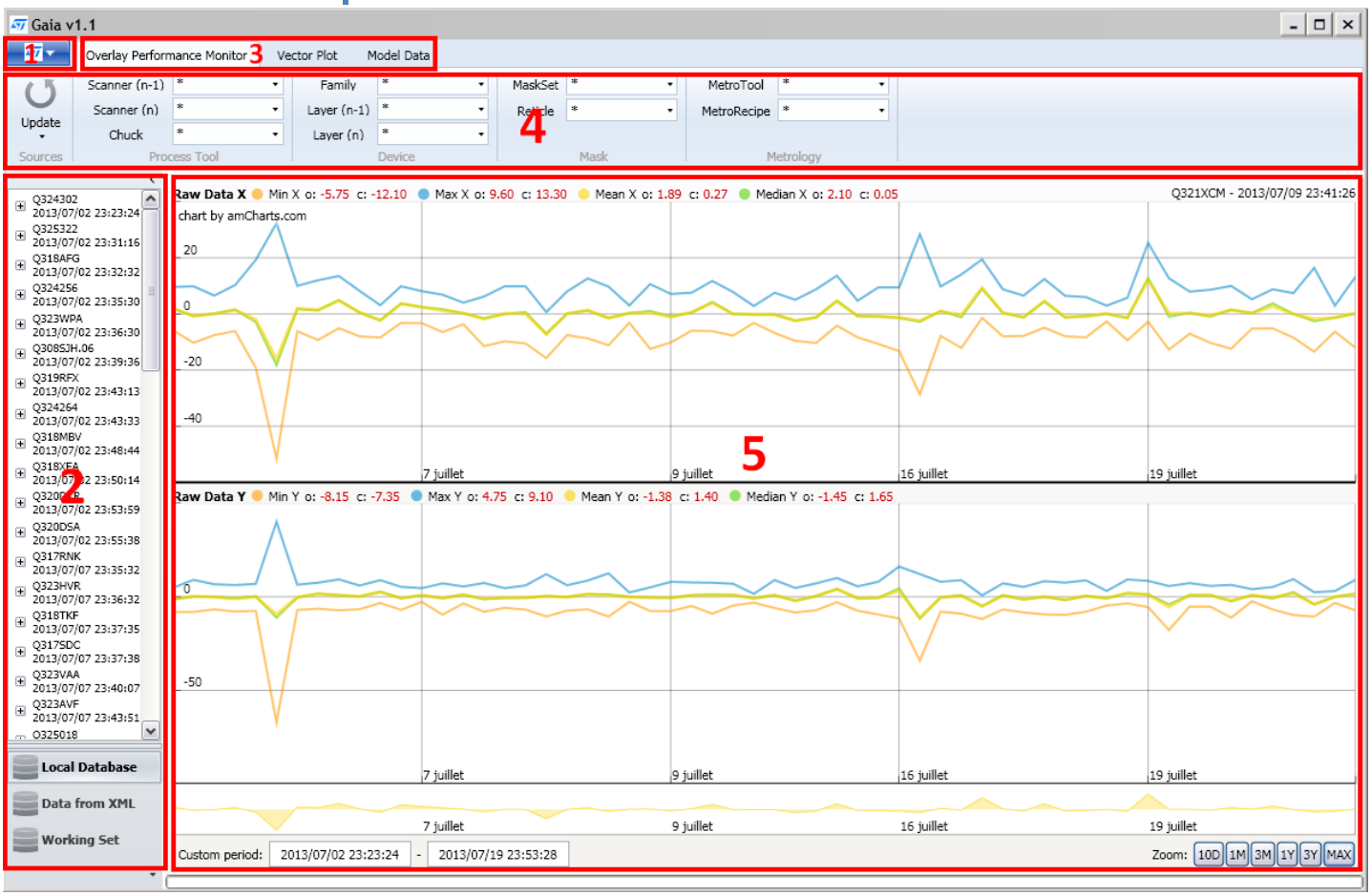

Figure 5 – Vue d'ensemble de Gaïa 1.1

Légende valable pour les deux figures :

1. Menu principal
2. Panneau de navigation / sélection des données
3. Sélection de la vue principale
4. Barre d'outils de la vue principale
5. Vue principale

Nous reviendrons plus en détail par la suite sur les changements de chaque partie définie ci-dessus, mais nous pouvons d'ores et déjà observé que l'ajout du ruban utilisé dans Microsoft Office a permis de gagner une place considérable en largeur. Le ruban pouvant être rangé d'un simple clic sur l'onglet ouvert, l'utilisateur peut si besoin regagner facilement la place perdue en hauteur. De même, le menu principal et les onglets de l'application sont maintenant sur la même ligne, ce qui fait encore gagner de l'espace en hauteur. Le menu de gauche permettant de sélectionner les données sur lesquelles l'utilisateur souhaite travailler est lui aussi nettement moins surchargé, même s'il a en conséquent perdu en fonctionnalités. L'interface en globalité est beaucoup plus claire.

III. 2. Gestion des données

III. 2. 1. Importation des données

La première étape pour utiliser Gaïa est l'importation du jeu de données sur lequel l'utilisateur souhaite travailler. Dans Gaïa version 0, elle se faisait par la sélection de fichiers XML en provenance de deux machines différentes : KLA-Archer ou YieldStar. C'est la première force de Gaïa, quel que soit le lieu de départ d'enregistrement des données, une fois importées dans l'application, les mêmes traitements pourront ensuite être effectués : toutes les données sont dans un format commun. Pour obtenir les données, il faut donc tout d'abord passer par les logiciels fournis par les équipementiers, KT-Analyser pour KLA-Archer et YieldStar pour la machine du même nom, qui vont permettre l'exportation des données de la machine en fichiers XML. Ses fichiers seront ensuite parsés par Gaïa pour permettre la récupération des données. C'est pour éviter ces intermédiaires qu'une des fonctionnalités majeures apportées par Gaïa version 1 est de passer outre en se connectant directement aux bases de données des équipements de la salle blanche pour avoir ainsi accès à des multitudes d'informations sans étape supplémentaire.

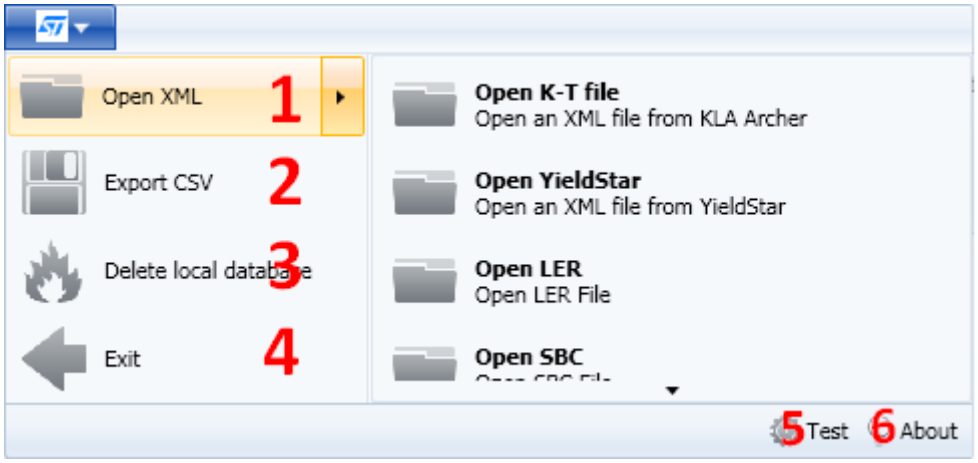

Figure 6 - Menu principal de Gaïa 1.1

1. Permet de charger des données dans l'application depuis des fichiers XML. Les fichiers peuvent être : Fichier KT (extrait depuis KT-Analyzer), fichier YieldStar, fichier LER et fichier SBC (ces deux derniers fichiers ne sont pas utilisables dans Gaïa v1. Ils permettaient d'appliquer des modèles SBC aux données dans Gaïa v0.)
2. Exporte l'intégralité des données de l'application dans un fichier CSV facilement importable dans MS Excel ou LibreOffice Calc
3. Supprime la base de données locale. L'application revient alors dans l'état neuf juste après l'installation (toutes les données et réglages sont perdus)
4. Quitte l'application
5. Lance le jeu de tests de l'application
6. Affiche la fenêtre à propos avec les informations de version et d'auteur

Aujourd'hui, Gaia version 1 est capable de se connecter directement à QArcher grâce à l'API fourni par KLA. Il n'est ainsi plus nécessaire de passer par KT-Analyzer, sélectionner les lots à étudier, de les exporter puis de les réimporter dans Gaïa. L'application propose les même filtres que ceux disponible dans KT-Analyzer, pour n'importer que les données qui nous intéressent : filtre sur l'identifiant des lots, sur la date, sur l'équipement, etc. L'importation se fait en un clic sur le bouton Update (voir 1 sur la figure 8 – Barre d'outils d'overlay performance monitor), la seule condition nécessaire est d'être connectée au réseau de l'entreprise.

Gaïa possède de plus un module pour se connecter à la « Dataware », une copie de la base de données de la salle blanche contenant toutes les informations dont l'utilisateur peut avoir besoin sur le contexte du traitement des lots : sur quelle machine de production et à quelle date et heure est-il passé, de même pour la machine de mesure, de quel manière a-t-il été exposé, quelle était la recette utilisée, le motif du masque, etc. Cette étape de connexion à la Dataware pour récupérer les informations sur le contexte du traitement est faite de manière systématique, que les lots soient importés par l'API KLA ou par des fichiers XML. Ces données sont utiles car les utilisateurs ne travaillent pas tous sur les même technologies, Gaïa permet donc à l'utilisateur de spécifier les données à afficher en sélectionnant le contexte qui l'intéresse (voir paragraphe sur le filtrage des données).

III. 2. 2. Panneau de sélection

Une fois les données importées dans l'application, elles deviennent disponibles dans le panneau latéral gauche. Ce panneau est constitué d'une treeview (2) qui affiche les données correspondant à la source sélectionnée (1). Les trois sources possibles sont « Database », qui contient les données importées à partir de l'API KLA, « XML » si les données ont été importées à partir de fichiers XML, ou « Working Set », qui contient les données sur lesquelles l'utilisateur est en train de travailler, pour lui permettre d'y accéder plus rapidement. « Working Set » contient aussi les données que l'utilisateur aurait pu créer à l'intérieur de Gaïa à des fins de simulation (voir paragraphe sur les Lots Virtuels). La treeview du panneau affiche les données de manière plus détaillée avec une arborescence Lot / Wafer / Mesure et une info bulle affichant les informations complètes au survol de la souris. Cela permet à l'utilisateur d'aller regarder une donnée précise s'il sait déjà ce qu'il veut étudier : il peut aller consulter directement une mesure s'il sait le lot, le wafer, le champ et la cible auxquels elle appartient. Voici à quoi ressemble le panneau de navigation :

1. **Source des données affichées**
   Il y a trois sources sélectionnables : les données importées dans l'application depuis KLA Archer, les données importées dans l'application depuis des fichiers XML et l'ensemble de travail courant. Les données de la source sélectionnée ici sont affichées dans la treeview au-dessus et dans le graphique de la vue principale à droite.
2. **Treeview des données**
   Cette treeview affiche les données sélectionnées selon trois niveaux de profondeur : le lot, le wafer ou la mesure. Les informations de bases sont affichées : ID et date de passage dans la machine pour le lot, slot pour le wafer, indice X, indice Y du champ et numéro de la target pour la mesure. Cliquer sur une donnée la plotera dans la vue vector plot view.
3. **Infobulle détaillée**
   Passer la souris sur un élément de la treeview fait apparaître une infobulle avec les informations détaillées sur cet élément.
4. **Bouton de minimisation du panneau de navigation**
5. **Options avancées pour le panneau de navigation**

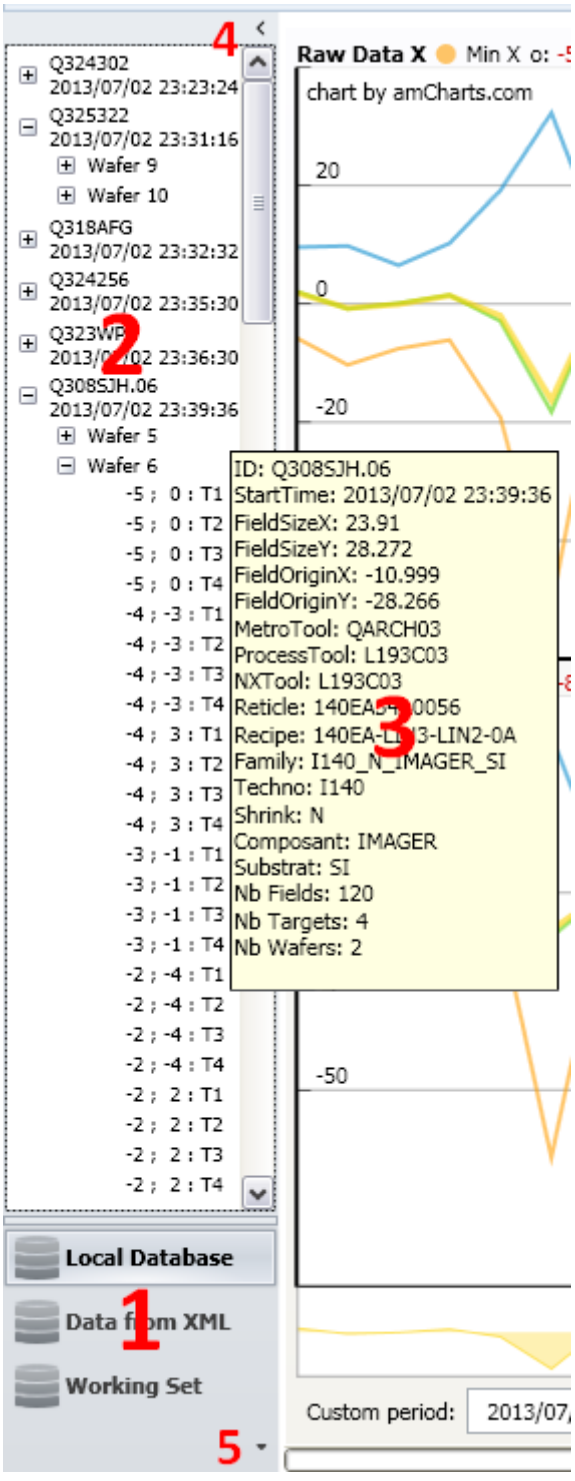

Figure 7 - Panneau de navigation de Gaïa version 1

III. 2. 3. Stockage des données

Une autre fonctionnalité clef apportée par Gaïa version 1 est l'intégration d'une base de données locale afin de stocker les données importées et les restaurer entre deux utilisations de Gaïa. Les prémices de cette fonctionnalité étaient déjà apparues dans Gaïa version 0 mais elle n'est devenue correctement utilisable qu'avec la version 1. Cette fonctionnalité est extrêmement importante car elle rend l'application utilisable dans un temps acceptable. En effet, importer des données est une opération très longue, qui peut prendre plus d'une minute par lot, le temps de récupérer les données dans KLA puis le contexte dans la Dataware. L'intérêt principal de Gaïa est de permettre l'observation des données sur de longue période (plusieurs semaines voire mois) ce qui représente des centaines de lots. Si à chaque redémarrage de Gaïa l'utilisateur devait importer les données des 6 derniers mois, le temps d'attente serait beaucoup trop important. Gaïa embarque donc une base de données interne dans laquelle sont systématiquement stockés les lots importés. A chaque redémarrage, Gaïa charge automatiquement les lots depuis cette base. Cette action est très rapide. Lorsque l'utilisateur ouvre l'application et veut mettre à jour les données, seules les données enregistrées par les équipements depuis la dernière ouverture de Gaïa doivent être récupérées à distance, les autres données sont déjà disponibles.

Une des contraintes de Gaïa étant l'installation sans les permissions d'administrateur, la base de données locale est une base SQLite, donc une base sous forme d'un fichier. Cela permet d'avoir à la fois les fonctionnalités d'une base de données (requêtes et interrogations précises, structure cohérente, etc) tout en ne nécessitant qu'une simple écriture dans un fichier et une librairie (DLL) supplémentaire empaquetée dans Gaïa. Une solution avec MySQL ou PostgreSQL aurait été beaucoup trop contraignante à déployer sur chacun des postes des utilisateurs, en demandant à chaque fois au service informatique d'installer le logiciel. Un avantage supplémentaire est que les utilisateurs peuvent facilement s'échanger les données de leur application : il leur suffit de se partager le fichier de la base de données. Un utilisateur ayant fraîchement installé Gaïa peut donc se retrouver avec toutes les données déjà présentes sans avoir à envoyer de requêtes aux équipements : il lui suffit de récupérer la base d'une personne utilisant déjà Gaïa !

III. 3. Le filtrage des données

Après l'étape d'importation ou de rechargement depuis la base de données locale, les données mesurées sont disponibles dans l'application. Cependant, une telle quantité de mesures est trop importante pour pouvoir être affichées telles quelles. Pour passer de la masse de données disponibles dans les bases aux mesures exactes qui intéressent l'utilisateur, des filtres sont disponibles à trois niveaux différents. Le premier est un filtre à l'importation. En utilisant l'accès par l'API, l'utilisateur cible les données qu'il souhaite importer dans la base de données locale (voir la fenêtre 2 dans la capture ci-dessous). De même pour les fichiers XML : l'utilisateur n'importe que les fichiers qu'ils souhaitent étudier. Une fois les données importées dans l'application, elles sont affichées sur un graphique qui permet d'avoir une vision globale des données en présentant pour chaque lot quelques indicateurs statistiques (moyenne, médiane, valeur maximum et valeur minimum).

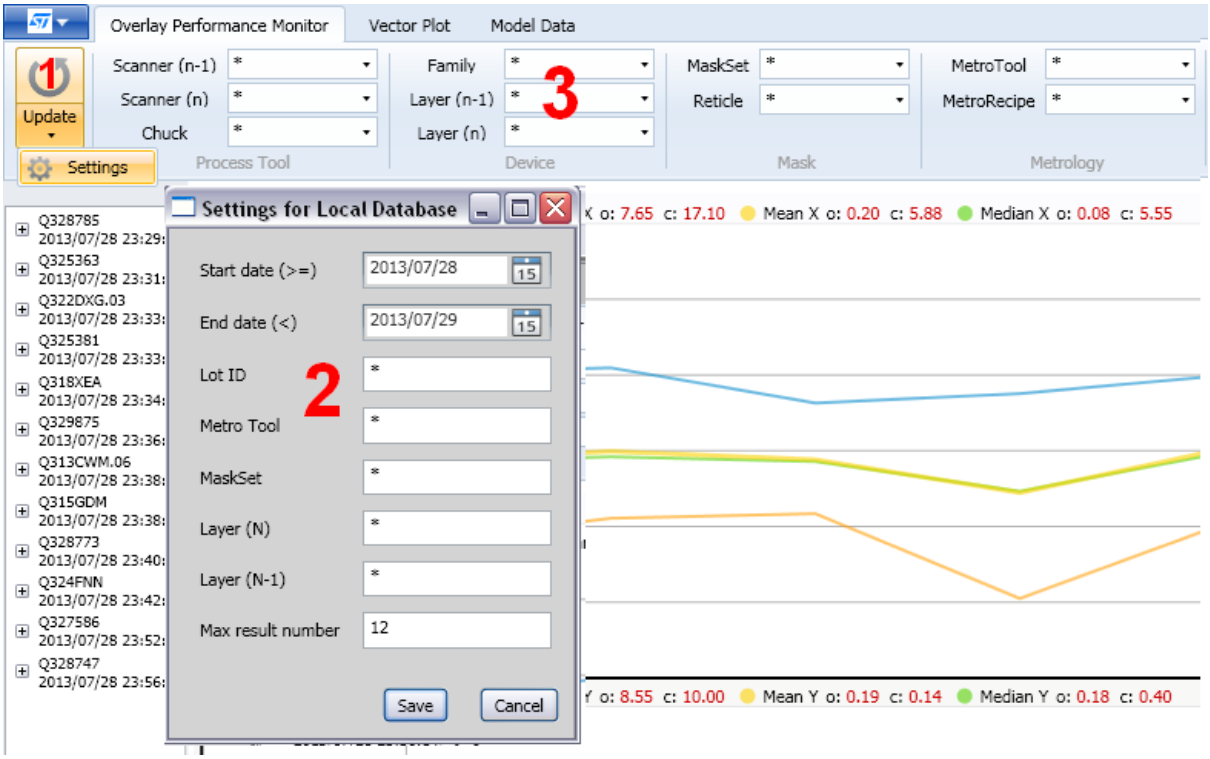

Figure 8 –Barre d'outils de la vue Overlay Performance Monitor

Le graphique affiche par défaut l'ensemble des données de l'application. L'utilisateur peut ensuite cibler les données affichées en sélectionnant une période précise sur le graphique, mais aussi en réglant les filtres plus précis disponibles dans le ruban pour modifier les données qui seront prises en compte dans le graphique (voir 3 sur la capture d'écran ci-dessus). Ces filtres se basent sur les données du contexte récupérées dans la Dataware. Ils proposent un filtrage plus précis car sur plus de critères que celui disponible à l'importation. Depuis le graphique, l'utilisateur peut ploter (c'est-à-dire afficher sur le plot) le lot qui l'intéressent. Le troisième niveau de filtre est accessible dans la vue Vector Plot View, il s'agit de filtrer les mesures affichées, sur leur emplacement avec le filtre géographique (uniquement le bord de la plaque, ou le centre, etc.) ou bien sur leur valeur : retrait des mesures supérieures à une valeur donnée, en prenant la valeur absolue de la mesure, son sigma, ou un pourcentage de mesures, par exemple ne garder que 95% (les 5% de valeurs les plus élevés

sont alors éliminées). Ces filtres permettent de s'intéresser à des données particulières, mais ils permettent aussi de retirer les valeurs jugées non pertinentes, souvent des erreurs de mesure. Ils sont facilement accessibles dans le bandeau de la vue Vector Plot View (voir figure 9 ci-dessous).

III. 4. L'affichage des mesures

Lorsque l'utilisateur plote les données, l'application passe automatiquement sur l'onglet Vector Plot View. Cet onglet est présenté dans la figure n°9. Il permet d'étudier en détail les données et de simuler l'application de modèles. Comme présenté dans « I. 3. Structure d'un wafer », nous retrouvons afficher sur le plot les champs, les cibles sur chaque cible un vecteur par wafer ploté représentant la mesure d'une erreur. Deux plots sont présents dans la vue, ce qui permet de comparer facilement l'impact du modèle. Par exemple, en affichant les données brutes sur le plot de gauche et le résiduel du modèle sur le plot de droite, l'utilisateur peut en un coup d'œil voir si le modèle a corrigé les erreurs de manière efficace. Les listes déroulantes permettant de définir quel type de mesure est affiché sur chaque plot sont réglées par défaut sur ces valeurs : données brutes à gauche et résiduel par wafer à droite. Grâce aux sélecteurs ou à la molette de sa souris, l'utilisateur peut facilement zoomer sur une partie spécifique du plot pour regarder plus précisément l'erreur. Il peut aussi varier l'échelle de manière proportionnelle pour accentuer les différences entre les vecteurs. La fonctionnalité sans doute la plus intéressante est enfin la possibilité de filtrer les données, comme décrit dans III. 3. Les mesures filtrées sont ensuite affichées en rouge. Un tableau récapitulatif des principales statistiques est affiché sous chacun des plots, indiquant à l'utilisateur la mesure minimum, maximum, la moyenne et l'écart à la moyenne, à chaque fois pour x et pour y.

1. Curseur pour régler le zoom (aussi modifiable avec Ctrl + molette) et l'échelle
2. Réglage du filtre géographique. La liste déroulante permet de choisir sur quel critère filtrer (pas de filtre, mesure dont l'emplacement est inférieur ou supérieur à la valeur indiquée), le champ permet de saisir la valeur du filtre et le bouton de l'appliquer.
3. Réglage du filtre sur la valeur. La liste déroulante permet de choisir sur quel critère filtrer (valeur absolue, sigma ou pourcentage), toutes les mesures supérieures sont enlevées. Le champ permet de saisir la valeur du filtre et le bouton de l'appliquer.
4. Permet de choisir quelles données sont affichées sur les plots (elles sont décrites dans I. 4.) : données brutes, modèle ou résiduel, par lot ou par wafer.
5. Plot affichant les données sélectionnées. Les mesures filtrées sont affichées en rouge.
6. Légende montrant les réglages choisis
7. Tableau récapitulatif des statistiques des données plotées

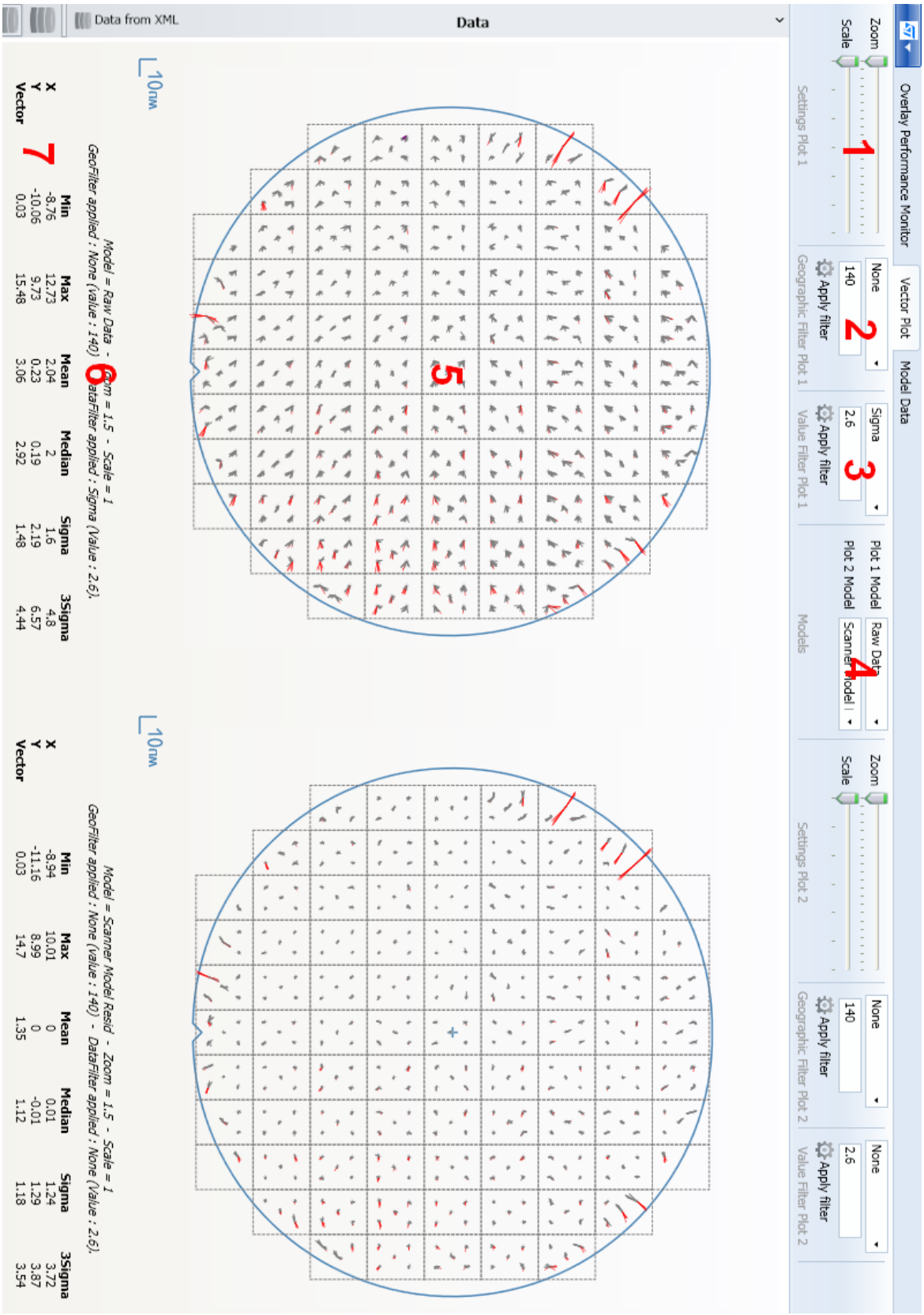

Figure 9 - Vue Vector Plot View

III. 5. Les lots virtuels

Afin de permettre une plus grande flexibilité sur les données que les simples filtres décrits ci-avant, Gaïa permet de générer des lots dits virtuels car ils n'ont pas d'existence réelle. Contrairement aux autres lots importés dans Gaïa qui sont présents dans la salle blanche, les lots virtuels ne servent qu'à faire des simulations à l'intérieur de Gaïa. Ces lots sont construits sur la base lot déjà existant dans l'application en récupérant la taille et les coordonnées des champs et des cibles. Ils sont modelés par l'utilisateur selon ses besoins : ce dernier peut effectuer des opérations arithmétiques entre des mesures et définir le résultat comme une mesure de son lot virtuel. Ces derniers ont après leur création le même statut qu'un lot normal, ils peuvent être affichés, corrigés selon différents modèles, et peuvent même servir de base pour créer un nouveau lot virtuel.

Lorsque l'utilisateur clique sur le bouton pour créer un lot virtuel, la fenêtre suivante s'ouvre (à gauche sur la figure n°10).

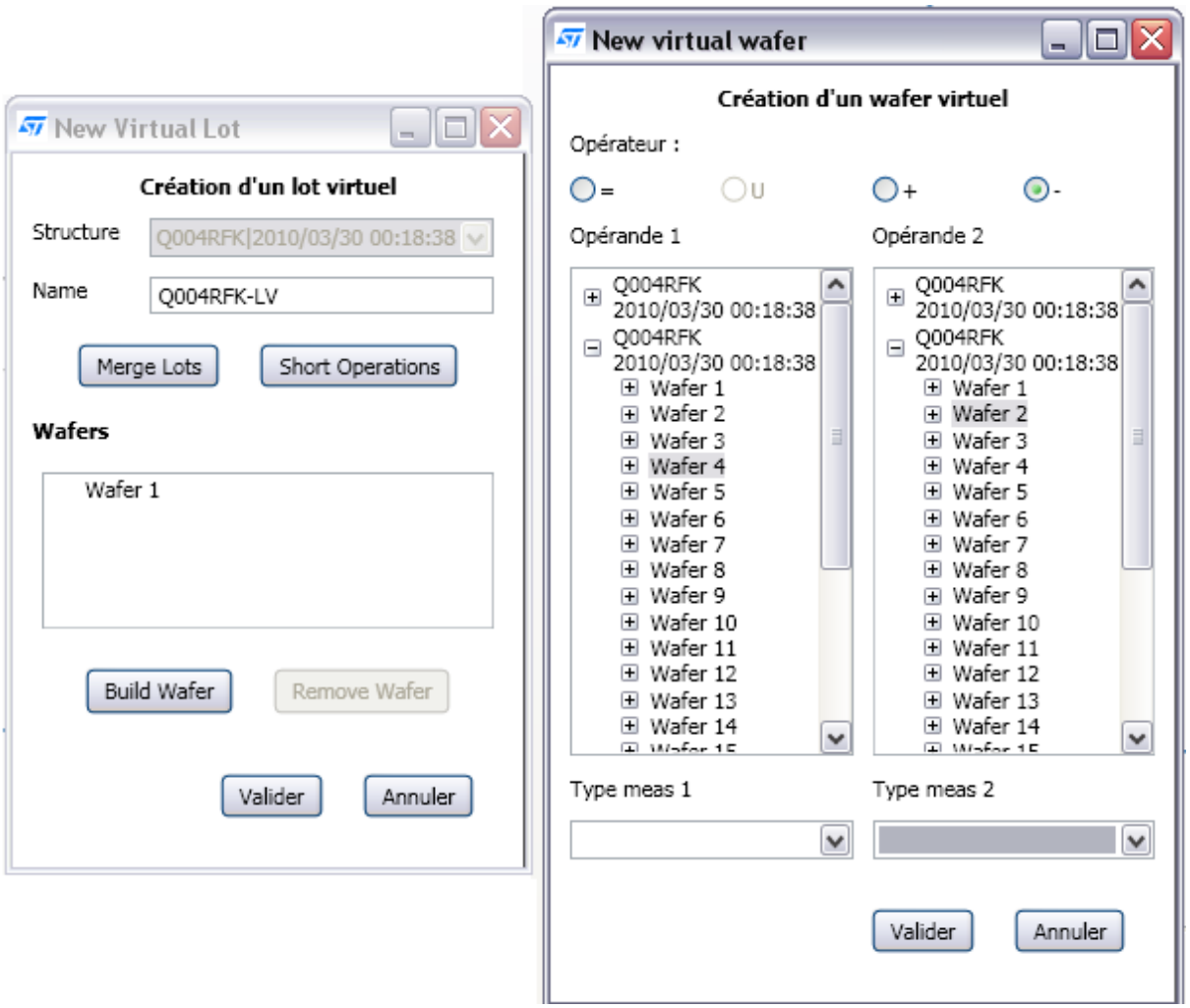

Figure 10 - Fenêtre de création d'un lot virtuel

Cette fenêtre propose plusieurs options à l'utilisateur, mais la première étape est de choisir le lot existant qui va servir de base pour le lot virtuel grâce à la première liste déroulante. Une fois choisi, le lot de base ne peut plus être changé car les données intégrées dans le lot virtuel ne pourront l'être que si leur layout (la disposition des cibles et des champs) est compatible avec celui du lot servant de base. Ensuite, l'utilisateur peut choisir le nom qu'il va donner au lot virtuel.

Trois choix s'offrent ensuite à l'utilisateur : deux sont des raccourcis et le troisième est la manière longue de créer un lot virtuel. Les deux raccourcis sont les boutons « Merge lots » et « Short Operations ». Ils permettent respectivement de créer rapidement un lot virtuel à partir de deux autres lots, pour le premier, en fusionnant les mesures (simplement en faisant la moyenne si deux mesures sont présentes pour la même plaque sur la même cible, ou de regrouper les mesures et les wafers dans un seul lot sinon), pour le second, de soustraire ou d'additionner les mesures de deux lots, en prenant chaque mesure correspondante, une à une. Short Operations est le mode le plus utilisé car il permet d'obtenir un nouveau lot représentant le résiduel d'un modèle. En effet, il suffit de faire la soustraction entre deux fois le même lot, les données brutes moins le modèle, pour obtenir le résiduel. Mais le résiduel sera cette fois un lot à part entière dans Gaïa, le résiduel sera présenté comme les données brutes, et il sera donc possible d'appliquer un autre modèle, avec d'autres filtres… Le troisième choix possible pour l'utilisateur est de construire le lot virtuel manuellement. Dans ce cas, l'utilisateur construit les wafers un par un, en prenant les deux wafers initiaux dans n'importe quel lot compatible, en choisissant à chaque fois le type de mesure et l'opération (voir la fenêtre de droite dans la figure n°10). Ce mode très générique permet de construire n'importe quel lot, au grès du besoin de l'utilisateur.

Les lots virtuels représentent donc un moyen puissant et générique pour créer des lots disponibles ensuite comme n'importe quel autre lot dans l'application, et donc susceptibles d'être eux aussi plotés, filtrés et simulé avec un modèle.

Conclusion

Gaïa est un projet mis en place pour apporter des solutions dans le domaine de l'overlay aux ingénieurs en photolithographie, qu'ils travaillent en recherche et développement ou à la maintenance quotidienne des machines. Il permet de regrouper dans un seul outil des données de provenances variées et de les visualiser facilement sur un graphique récapitulatif, même si les données s'étalent sur plusieurs mois. Grâce à trois niveaux de filtres, l'utilisateur de Gaïa peut cibler facilement les données qui l'intéressent pour les étudier plus en détail. Le logiciel propose ensuite la possibilité d'appliquer des modèles de correction pour tester les améliorations possibles à apporter à la machine. Avec la notion de lots virtuels, Gaïa va encore plus loin et permet de faire des simulations jusqu'à présent impossibles.

J'ai développé seul ce logiciel pendant mes trois années d'alternance pour en faire un prototype démontrant l'existence d'un besoin. Cela m'a permis de découvrir toutes les facettes de la création d'un projet, de l'analyse du besoin au déploiement. Cependant, il y avait aussi quelques côtés négatifs, comme l'absence de retours techniques sur mon travail ou l'impossibilité d'avoir certains outils par absence de budget.

Le développement a été mené de manière incrémentale, avec une nouvelle itération par période en entreprise. Ce rythme s'est très bien marié avec mon alternance à l'école mais il a surtout été possible grâce à la flexibilité du développement et à la proximité des utilisateurs. Il était très agréable de travailler en contact direct avec eux.

La version une de Gaïa développée pendant ce Projet de Fin d'Étude est l'aboutissement de ces trois années de travail. Avec un accès direct aux données de la salle blanche et une interface utilisateur complètement revue, elle corrige les deux plus gros manques de Gaïa version zéro et permet à présent de travailler sur des milliers de données étalées sur de longues périodes. J'ai été très heureux de travailler sur ce projet passionnant pendant ces trois années et j'espère qu'il sera autant utile à ses utilisateurs qu'il a été intéressant à créer.

Remerciements

Ce rapport et la soutenance qui le suit sont l'aboutissement non seulement d'un Projet de Fin d'Étude de sept mois mais aussi d'une alternance de trois ans entre l'Ensimag et STMicroelectronics. Ces années ont été riches en expérience tant à l'école qu'en entreprise, et je souhaitais remercier ici les personnes qui m'ont accompagné durant l'ensemble de mon cursus. Du côté de l'école, je remercie pour leur accompagnement Mme Brigitte Plateau pour ma première année d'alternance suivie par Mme Joëlle Thollot pour les deux années suivantes en tant que tutrices pédagogiques. L'école joue un rôle important dans l'alternance d'un apprenti même dans ses périodes en entreprise, et leurs indications et remarques ont été d'une aide précieuse. Pour la partie entreprise, je remercie mon manageur Mr Philippe Moschini qui m'a immédiatement considéré comme un membre de l'équipe à part entière. Je remercie aussi Mr Maxime Gatefait, utilisateur précurseur de mon application dont les retours et les suggestions ont permis au projet de faire de belles avancées. Et bien sûr, je remercie mon maître d'apprentissage Mr Auguste Lam, à l'initiative de ce projet et avec qui j'ai travaillé pendant ces trois années.

Lexique

Cluster : Equipement présent dans la salle blanche qui permet plusieurs étapes de la fabrication des plaques par lithographie. Il est la réunion de la track et du scanner.

Field : ou champ en français. Zone rectangulaire du wafer. Chacun des nombreux composants gravés sur le wafer est dans un field séparé. (voir figure 2)

Lithographie : Technique de gravure basée sur la réaction d'une résine à la lumière. Les composants sont donc dessinés en exposant uniquement les zones souhaitées.

Mesure : Une mesure overlay est une mesure optique effectuée sur les targets entre le niveau exposé et un niveau précédent de référence.

Modèle : appliqué aux mesures, il a pour but de les corriger au maximum

Overlay : La mesure d'overlay est une mesure optique entre le niveau actuel exposé et un niveau précédent de référence. C'est donc l'erreur d'alignement existante entre deux couches.

Scanner : partie du cluster dans laquelle la plaque enduite de résine est exposée à la lumière à travers un masque permettant d'appliquer un motif spécifique.

Target (ou cible) : Indicateur marqué sur la plaque pour permettre de contrôler sa position. La principale mesure effectuée est la différence entre l'emplacement estimé et l'emplacement mesuré de la cible.

Track (ou piste) : partie du cluster qui prépare la plaque avant d'entrer dans le scanner, notamment en l'enduisant de résine photo-sensible et en la passant dans un four.

Wafer : plaque de silicium sur laquelle sont gravés les composants électroniques.

Layout : Disposition et taille des champs et des cibles sur la plaque

Table des figures

Résumé

STMicroelectronics est une entreprise franco-italienne spécialisée dans la fabrication de composant semi-conducteur. Le processus utilisé par STMicroelectronics pour dessiner les circuits imprimés sur les plaques de silicium avec une très grande précision est appelé photolithographie. Un circuit est réalisé par superposition de couches qui doivent être alignées le mieux possible afin d'éviter un composant non fonctionnel. L'erreur d'alignement entre les couches est désignée par le terme d'overlay. Le but du projet Gaia est de fournir un outil informatique aux ingénieurs de lithographie pour les aider à minimiser l'overlay et son impact. Pour cela, l'application est capable d'importer de nombreuses données depuis des sources variées et de les visualiser rapidement grâce à des graphiques et des indicateurs statistiques. Il est ensuite possible de cibler une donnée particulière grâce à différents filtres et de l'étudier plus précisément par exemple en calculant un modèle qui tente de minimiser l'erreur en proposant des corrections. Durant mes trois années d'alternance, j'ai mené le projet Gaïa de l'analyse du besoin au déploiement en passant par la conception / spécification et le développement.

Mots-clefs :
Microélectronique, Photolithographie, Overlay, Data-Mining, Optimisation, Analyse, Statistiques

Summary

Keywords :

Microelectronic, lithography, overlay, data-mining, optimization, analyze, statistics