

**Projet de Fin d'Étude  
Rapport Final**

**« Recherche de méthode d'estimation de volume de production à risque »**

Équipe 5<sup>ème</sup> Année : Team-War  
Jaafar AMRANI-MESBAHI  
Fabien GARCIA  
ABDELALI NAIT BELKACEM  
Rahma NAKARA  
Philippe NGUYEN



Team  
**W@R**

Génie Industriel et Informatique

*Tuteur :*

Claudia Frydmann

Lundi 24 Janvier 2011

[Team-war@prunetwork.fr](mailto:Team-war@prunetwork.fr)



## Table des matières

Table des figures	6
Chapitre 1. Introduction	7
<b>partie 1. Contexte de l'étude</b>	8
Chapitre 2. Présentation de STMicroelectronics	9
2.1. Présentation de STMicroelectronics	9
Chapitre 3. Cahier des charges	11
3.1. Formalisation du cahier des charge à partir du besoin du client	11
3.2. Mission du PFE	11
3.3. Plan de l'étude	12
<b>partie 2. Etat de l'art</b>	13
Chapitre 4. Conférences	14
4.1. Operational risk evaluation and control plan design	14
4.2. A novel approach to minimize the number of controls in the defectivity area	14
Chapitre 5. Méthode $D^3$	16
5.1. Généralités	16
5.2. Objectifs de la méthode	16
5.3. Hypothèses de la méthode	16
5.4. L'algorithme	17
5.5. Algorithme détaillé	17
Chapitre 6. An aproach for operational risk evaluation and its link to control plan	20
6.1. Description de la méthode	20
6.2. Résultats	22
6.3. Remarques	23
Chapitre 7. Optimizing Return on inspection trough defectivity smart sampling	24
7.1. Description de l'algorithme	24
7.2. Résultats	26
7.3. Remarques	26
Chapitre 8. Utilisation du jugement d'expert	27
8.1. Utilisation de l'avis d'experts	27
8.2. AMDEC	27
8.3. Démarche	27
8.4. Objectif de démarche	28

8.5. Avantages	28
8.6. Inconvénient	28
<b>Chapitre 9. Optimized Design of Control Plans Based on Risk Exposure and Resources Capabilities</b>	
9.1. Introduction	30
9.2. Description de l'approche	30
9.3. Remarque	31
<b>Chapitre 10. Computation of wafer-At-Risk from theory to real life demonstration</b>	
10.1. Description de l'algorithme	32
10.2. Résultats	32
10.3. Remarques	33
<b>Chapitre 11. Analyse du risque</b>	
11.1. Méthode générale	35
11.2. Gravité du dommage (G)	35
11.3. Fréquence ou durée d'exposition au phénomène dangereux (F)	35
11.4. Probabilité d'occurrence de l'événement dangereux (O)	36
11.5. Possibilité d'évitement du dommage (P)	36
11.6. Réduction du risque	36
11.7. Élimination des phénomènes dangereux et réduction du risque :	37
<b>Chapitre 12. Bilan sur l'état de l'art</b>	
12.1. La démarche AMDEC	39
12.2. L'étude de la gestion du risque	39
12.3. Plan de contrôle optimisé en se basant sur le risque	39
12.4. An Approach for Operational Risk Evaluation and its Link to Control Plan	40
12.5. Optimizing Return On Inspection Through Defectivity Smart Sampling	40
12.6. Computation of Wafer-At-Risk from Theory To Real Life Demonstration	40
12.7. Operational risk evaluation and control plan design	40
12.8. La méthode $D^3$	40
12.9. Conclusion de l'état de l'art	41
<b>partie 3. Développement de la solution</b>	42
<b>Chapitre 13. Algorithme du marcheur</b>	43
13.1. Introduction	43
13.2. Description de l'algorithme	43
13.3. L'algorithme	44
13.4. Spécification du Code	46
13.5. Implémentation du marcheur	48
13.6. Résultats	50
13.7. Inconvénients de cette méthode	52
<b>partie 4. Conclusion de l'étude</b>	53
<b>Chapitre 14. Conclusion</b>	54
<b>partie 5. Annexes</b>	55

<b>Chapitre 15. Code Source</b>	<b>56</b>
<b>15.1. Classe Batch</b>	<b>56</b>
<b>15.2. Classe Workstation</b>	<b>59</b>
<b>15.3. Classe Marcheur</b>	<b>63</b>
<b>Bibliographie</b>	<b>66</b>
<b>Bibliographie</b>	<b>66</b>

## Table des figures

2.1.1 Principaux sites de production	9
2.1.2 Historique du site de Roussent	10
2.1.3 Divers groupes de produit	10
5.1.1 Schéma de la situation traité par $D^3$	16
6.1.1 Added value of a control plan	20
6.1.2 Potential Loss when no control plan	21
6.1.3 Potential Loss for a control with one control	21
6.2.1 Examples of risk evolution for different control plans	22
7.1.1 The production system	24
7.1.2 Risk reduction variation	25
7.1.3 Wafer-At-Risk prototype	25
7.1.4 Zoom on case of skip	26
8.3.1 Tableau récapitulatif de la démarche AMDEC	29
9.2.1 Approche proposée	31
10.1 Flag algorithm	32
10.1 Boundary conditions	33
10.1 Global W@R and W@R by recipe	33
11.1 Les éléments du risque	35
11.6 Réduction du risque	37
13.3 Modèles d'évolution fiabilité (ou la criminalité) d'une machine (ou quartier)	45
13.4 Diagramme de classe des objets entités	48
13.5 Diagramme de classe des différentes entités	49
13.5 Diagramme de classe des constantes	50
13.5 Diagramme de classe des extractors	50
13.6 Représentation des résultats après calculs	51
13.6 Représentation des résultats après calculs	52

## CHAPITRE 1

# Introduction

Le domaine de la micro-électronique est extrêmement compétitif. Un des points clefs pour être compétitif est la réduction de la non qualité dans la production des lots de *chips* électroniques.

Toute la difficulté du problème tient du fait que le contrôle des produits est extrêmement cher et très long. Il faut donc trouver un moyen d'optimiser les contrôles de *chips* et des machines pour contrôler les produits les plus risqués.

Cette étude consiste à déterminer la faisabilité pour la mise en place d'une méthode pour optimiser le contrôle de la qualité.

Le présent rapport sera organisé de la façon suivante :

- Tout d'abord nous préciserons un peu plus finement les contraintes et le contexte de la micro-électronique,
- Ensuite nous nous intéresserons à l'état de la connaissance dans ce domaine ainsi qu'à un Benchmarking, afin de trouver un moyen utilisé dans les autres industries pour résoudre ce problème,
- Finalement l'absence de solution concrète nous a poussé à développer notre propre solution, nous détaillerons cette solution et discuterons de sa pertinence.



## Première partie

### Contexte de l'étude



## CHAPITRE 2

# Présentation de STMicroelectronics

## 2.1. Présentation de STMicroelectronics

STMicroelectronics[1] (plus connu sous le nom de ST) est une multinationale du semi-conducteurs, créée en 1987, suite à la fusion de SGS Microelettronica et de Thomson Semiconducteurs, sous le nom de SGS-THOMSON. Renommée en STMicroelectronics, après le retrait de Thomson en 1998. Cette entreprise développe, fabrique et commercialise des puces électroniques à travers le monde (Cf Fig2.1.3).

Elle se décompose comme suit :

- La société mère STMicroelectronics est de droit hollandais (enregistrée à Amsterdam),
- la direction administrative est regroupée en grande partie sur le site de Genève en Suisse,
- la direction opérationnelle du groupe est italienne, implantée sur le principal site du groupe à Agrate (à côté de Milan).

Société en expansion économique depuis sa création, son chiffre d'affaire a même doublé en dix ans depuis son nouveau nom « STMicroelectronics » (CA : \$9,84 milliards en 2008 contre \$4,25 milliards en 1988).

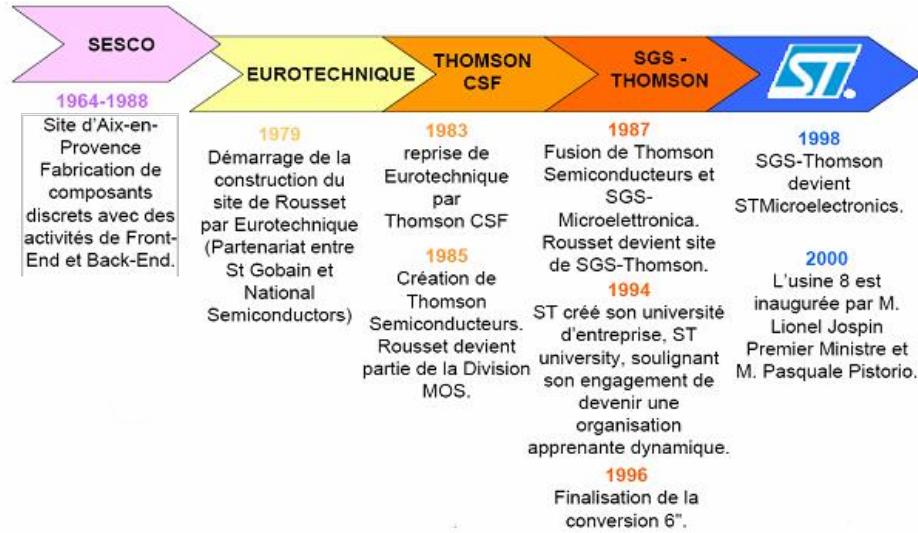
S'appuyant sur sa technologie et son expertise, ST se positionne au 5<sup>e</sup> rang mondial dans la fabrication des semi-conducteurs depuis 2005. 51000 employés (approximativement) sont répartis dans plus de 15 sites de production (cf Fig 2.1.1), plus de 39 centres de R&D et plus de 78 bureaux de vente dans le monde.

FIGURE 2.1.1. Principaux sites de production



Cette carte indique la répartition des usines de production de ST à travers le monde

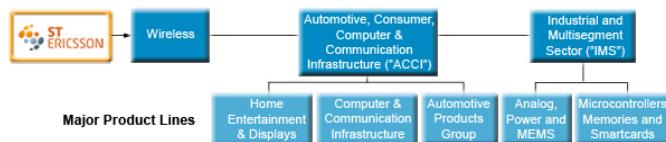
FIGURE 2.1.3. Historique du site de Rousset



*Le schéma montre toutes les phases des changements d'actionnariat. Historique du site de Rousset.*

ST propose divers groupes de produit, chaque groupe étant lui-même composé de plusieurs divisions qui gèrent de façon autonome la conception, la fabrication et la commercialisation de leurs produits. (cf Fig 2.1.2) :

FIGURE 2.1.2. Divers groupes de produit



## CHAPITRE 3

### Cahier des charges

« Le travail consistera en une étude de faisabilité sur l'évaluation des 'wafer@risk' (WAR). Actuellement nous comptons le nombre de plaque de silicium travaillées entre deux contrôles (mesures). Ce comptage peut être référencé par rapport à une machine de production ou par rapport à une technologie. Ce comptage ne tient pas compte de l'existence de plusieurs type de paramètres ni de l'interaction potentielle entre paramètres. Pour l'instant les mesures effectuées sur des plaques de production et les mesures sur plaque de test liées aux qualifications périodique sont prisent en compte de la même manière sans tenir compte de la signification de la mesure. Cette méthode ne nous permet pas de régler nos outils d'échantillonnage par rapport à un risque maximal garanti. »

#### 3.1. Formalisation du cahier des charge à partir du besoin du client

La pratique actuelle de STMicroelectronics consiste à compter le nombre de plaques de silicium travaillées entre deux mesures. Ce comptage peut être référencé à une machine de production ou à une technologie. Par contre, il ne prend pas en compte de l'existence de plusieurs type de paramètres ni de l'interaction potentielle entre paramètres.

Leur cycle de production moyen est de trois mois, tracé par un système GPAO. La production est de type « jobshop<sup>1</sup> » par lots de 25 plaques de silicium. Elle ne suit pas le principe FIFO<sup>2</sup> et ce qui constitue une des difficultés majeure de cette étude.

Il existe deux types de contrôle pour assurer les volumes de production à risque :

- Contrôle de qualité sur des plaques de silicium<sup>3</sup> travaillées à l'aide des appareils de métrologie. Ce type de contrôle est très onéreux et nécessite plusieurs heures pour seulement 3 plaques.
- Contrôle de fiabilité des machines de production, c'est une tâche de maintenance périodique consistant à vérifier son bon fonctionnement à l'aide d'une plaque de silicium vierge. Ce type de contrôle n'est ni onéreux ni long mais engendre l'indisponibilité de la machine .

Les règles de contrôle sont basées sur l'expérience des contremaîtres, sans procédure définie. Cette méthode ne permet donc pas de pouvoir régler les outils d'échantillonnage par rapport à un risque maximal garanti.

#### 3.2. Mission du PFE

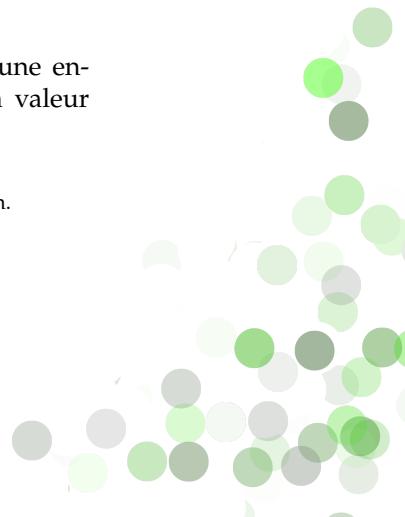
Le coût de la non-qualité est souvent l'ancre qui peut faire couler une entreprise. D'une part, par son coût important qui ne rentre pas dans la valeur ajoutée, et puis aussi par son impacte sur le marché concurrentiel.

---

1. Production en atelier, mouvement brownien des produits dans le cycle de production.

2. First In First Out

3. Contrôle du produit par échantillonage



Notre travail consiste à rechercher une ou des solutions susceptibles d'améliorer la gestion des contrôles et de pouvoir maîtriser la production à risque.

La mission telle qu'elle est mentionnée dans le cahier des charges du client est la suivante : « Le travail consistera en une étude de faisabilité sur l'évaluation des 'wafer@risk'<sup>4</sup> ».

En principe, notre mission se limite à la recherche des solutions et l'étude de leur faisabilité. Étant donné le domaine d'étude risque d'être limité en source d'information due à la confidentialité et à la spécificité du domaine d'activité (micro-électronique). Nous serons peut-être amener à imaginer notre propre solution.

### 3.3. Plan de l'étude

Une étude préliminaire pour balayer le sujet et les pistes de recherche d'information possibles. Ce qui nous amène à l'étude de l'art, une sorte de travail de *Benchmarking*. A partir du bilan de cette étude, de fouille d'information, nous essaierons de trier et de garder les solutions pertinentes pour une étude plus approfondie. Dans le cas où l'état de l'art ne permet pas de répondre pertinemment à nos attentes. Une tentative solution sera développer par notre équipe pour répondre à notre problème d'optimisation, suivi de la phase de simulation et de validation de la solution.

---

4. « Wafers At Risk » (W@R) est un indicateur qui représente le nombre de plaques traitées sur un équipement depuis la dernière tâche de contrôle.

## **Deuxième partie**

### **Etat de l'art**



## CHAPITRE 4

# Conférences

### 4.1. Operational risk evaluation and control plan design

Ce document est un support de présentation de Belgacem Bettayeb, qui présente sa thèse « An approach for operational risk evaluation and its link to control plan » en collaboration avec P. Vialletelle, S. Bassetto et M. Tollenaere.

Cette approche d'évaluation des risques est basée sur l'évolution du « Potential Loss » (défectivité potentielle). Cette évolution est supposée linéairement croissante avec le nombre d'exécution (traitement par la machine), sans contrôle, sur un horizon donné. Ce risque potentiel est réinitialisé, à une valeur seuil, après chaque tâche de contrôle ou de maintenance. L'évolution du « Potential Loss » dépend de la gestion (planification) des tâches de qualité et de maintenance. De plus s'ajoute à cela une fonction d'évaluation de la valeur ajoutée des tâches de contrôles mises en place afin de déterminer la bonne planification de ces tâches pour un résultat optimal du risque.

Cependant ce support de présentation n'apporte pas plus d'information pertinente que l'extrait de la thèse pour être retenu comme solution de notre problème.

### 4.2. A novel approach to minimize the number of controls in the defectivity area

Cette présentation expose une nouvelle approche de gestion des tâches de contrôle, issue des travaux de J. Nduhura Munga, S. Dauzère-Pérès, C. Yugma et P. Vialletelle.

Cette nouvelle approche a pour objectif :

- d'évaluer les contrôles en défectivité.
- de déterminer les processus de fabrication qui sont sur-contrôlés ou sous-contrôlés
- de pouvoir maîtriser dynamiquement les risques<sup>1</sup> sur les divers équipements de production par des contrôles en défectivité.
- de pouvoir réduire les contrôles inutiles en évitant « intelligemment » les produits qui n'apportent pas d'information pertinente.

Cette étude repose sur l'indicateur « Wafers At Risk » (W@R), qui représente le nombre de plaques traitées sur un équipement depuis la dernière tâche de contrôle. Cet indicateur sera exploité par un algorithme de calcul des « Permanent Index per Context (PIC) ». Un PIC est une sorte de compteur qui s'incrémenter à chaque fois que le « context » est vérifié et se réinitialise lors d'un événement spécial (maintenance préventive ou tâche de contrôle). Le « context » peut être un équipement, une cellule ou une recette.

L'implémentation de cet algorithme sur une ligne de production, nécessite de définir au préalable certains produits qui vont être contrôlés lors de la première exécution et qui permet d'atteindre les objectifs définis précédemment.

---

1. nombre de produits traités sur un équipement entre deux contrôles en défectivité.

Les résultats de l'implémentation de l'algorithme sur une ligne de production montrent, au bout de deux semaines, une réduction de 35% des produits à contrôler sans incident sur le taux de défectivité de la production.

Malheureusement, nous aboutissons à la même conclusion que pour la thèse de « Optimizing return on inspection through defectivity smart sampling » qui traite plus ou moins la même approche. Les résultats sont intéressants mais incomplets. De plus cette approche ne permet pas réduire les risques en défectivité mais seulement l'échantillonnage pour la mesure. Cette la réduction de la population des échantillons réduit par conséquence les coûts des tâches de qualité.



## CHAPITRE 5

# Méthode $D^3$

### 5.1. Généralités

Cette algorithme[2] consiste à regarder à la sortie tous les produits (ayant des chemins de productions différentes) et déterminer quels sont les points dans la production qui sont le plus sensible. L'idée est d'observer à la fin les produits finis et de déterminer les opérations à risques par analyse statistique des défauts.

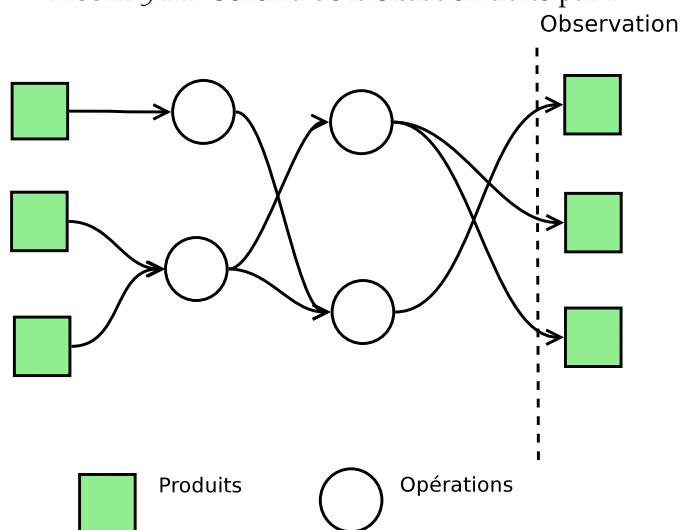
### 5.2. Objectifs de la méthode

Objectif minimiser le nombre maximum de test pour identifier tous les objets défectueux.

### 5.3. Hypothèses de la méthode

- Il n'y a que 2 état pour le produit *defective* ou *good*.
- Chaque test se fait sur un sous ensemble  $n$  objets et n'ont que deux possibilités *positive* ou *negative*.
- Un résultat *positive* indique que l'échantillon (ou sous ensemble) contient au moins un élément *defective*
- Un résultat *negative* indique que l'échantillon est pur et qu'il ne contient que des éléments *good*.
- Deux type de méthodes :

FIGURE 5.1.1. Schéma de la situation traité par  $D^3$



Les produits issus de la production proviennent de différents chemins l'analyse des produits finis permet de connaître les opérations sensibles grâce à la méthode  $D^3$

- Stochastique : les objets ont une probabilité d'être *defective* et le but est de diminuer le nombre de test attendu pour identifier tous les éléments objet *defective*.
- Déterministe : le nombre d'éléments *defective* est connu à l'avance, l'objectif est de minimiser le nombre maximal de test pour identifier tous les éléments *defective*.
- $M(n, d)$  nombre optimal de test optimal pour identifier les  $d$  éléments *defective* quand on connaît le nombre  $d$  parmi  $n$  éléments.
- $T_A(n, d)$  nombre de test requis par l'algorithme  $A$  pour identifier parmi  $n$  éléments tous les  $d$  *defective* élément quand on connaît  $d$  à l'avance.
- Si  $d = 0$  l'algorithme effectuera  $\log(n)$  tests inutile. Dans ce cas on vérifiera les éléments inconnus.
- Donc  $A$  est  $\alpha$ -compétitif si pour  $0 \leq d \leq n$ ,  $T(n, d) \leq \alpha M(n, d) + \beta$  avec  $\alpha = 2,75[5]$  et  $\beta$  une constante et un algorithme de ce genre existe.

#### 5.4. L'algorithme

L'algorithme utilise la stratégie suivante. Il teste des collections disjointes d'item de taille  $1, 2, \dots, 2^i$  jusqu'à ce qu'il trouve un élément *defective*.

À cette étape l'algorithme a détecté  $1 + 2 + 3 + \dots + 2^{i-1} = 2^i - 1$  bon éléments et a trouvé une collection contaminée de taille  $2^i$ . En utilisant une recherche binaire l'élément peut-être détecté par  $i$  tests additionnels.

L'algorithme effectuera  $i + 1$  tests, donc pour le prix de  $2i + 1$  tests l'algorithme apprend l'état de  $2^i$  objets.

En d'autre mots l'état de  $a$  nouveaux éléments est connu en effectuant  $2 \log a + 1$  tests.

#### 5.5. Algorithme détaillé

**Algorithme DOUBLE.** L'algorithme est présenté de manière détaillée dans les algorithme 1, 3, 2 et 4



**Algorithm 1** Algorithme Double pour tester  $n$  objets

---

**Algorithm Double**

```

U:= {1 ,..., ,n}
D:=o;
G:=o;
While |U|>= 3 do
    {x,y,z}:= 3 objets arbitraire de U
    TEST({x,y,z});
    if positive 3-TEST({x,y,z});
    if negative
        U:=U-{x,y,z};
        TEST(U);
        if negative
            G:=G+U;
            U:=o;
        if positive
            k:=4;
            repeat
                C:=k arbitraire de U ou U si k>=|U|;
                TEST(C);
                if positive
                    x:=BINARY-TEST(C);
                    D:=D+{x };
                    U:=U-{x };
                    abort-repeat;
                if negative
                    k:=2k;
                    G:=G+C;
                    U:=U-C
            end-repeat
        end-While
        FINAL-TESTS;
    end-algoritm

```

---



**Algorithm 2** Procédure  $\beta$ -TEST

---

```

Procedure  $\beta$ -TEST( $\{x, y, z\}$ )
    TEST( $\{x\}$ );
    if positive
        D:=D+ $\{x\}$ ;
        TEST( $\{y\}$ );
        if positive D:=D+ $\{y\}$ ;
        if negative G:=G+ $\{y\}$ ;
        U:=U− $\{x, y\}$ ;
    if negative
        G:=G+ $\{x\}$ ;
        TEST( $\{y\}$ );
        if positive
            D:=D+ $\{y\}$ ;
            U:=U− $\{x, y\}$ ;
        if negative
            D:=D+ $\{z\}$ ;
            U:=U− $\{x, z\}$ ;

```

---

**Algorithm 3** Algorithme Double pour tester  $n$  objets

---

```

Procedure BINARY-TEST(C);
    k:=|C|;
    repeat
        C':=|k/2| éléments arbitraire de C
        TEST(C');
        if positive C:=C';
        if negative C:=C−C';
        k:=|C|;
    until k=1;
    x:= le seul élément de C;
    return (x);
end-procedure;

```

---

**Algorithm 4** Procédure FINAL-TESTS

---

```

Procedure FINAL-TESTS
    while U!=o do
        x:= un élément arbitraire de U;
        TEST( $\{x\}$ );
        if positive D:=D+ $\{x\}$ ;
        if negative G:=G+ $\{x\}$ ;
        U:=U− $\{x\}$ ;
    end-while;
end-procedure;

```

---



## CHAPITRE 6

# An approach for operational risk evaluation and its link to control plan

### 6.1. Description de la méthode

Cet article[4] présente une méthode d'évaluation des risques opérationnels aidant à la prise de décisions sur les plans de contrôle. La méthode consiste à évaluer l'évolution du risque  $R^0(t)$  sur un horizon (durée) considéré ( $H$ ) sans aucun contrôle. Puis, une valeur ajoutée (en termes de risques) pourrait être évaluée avec un plan de contrôle X (cf. Fig 6.1.1).

Le risque est calculé comme le produit de la probabilité de l'évènement non-désiré (NDE : panne, défaillance, etc) et de la Perte Potentielle (PL) lorsque cet événement se produit. La Perte Potentielle est ici exprimée par le nombre de produits perdus (puce, plaquette ou le lot). Lorsqu'un événement non-désiré se produit, la qualité de ses prochaines exécutions (machine) sera alors diminuée jusqu'à son prochain contrôle (le contrôle signifie mesurer et corriger si ce n'est pas "OK"). Dans ce cas, la Perte Potentielle augmentera linéairement avec le nombre d'exécutions.

En l'absence du plan de contrôle, l'évolution de la Perte Potentielle dépend du moment d'apparition de l'évènement non-désiré (cf. Fig 6.1.2).

FIGURE 6.1.1. Added value of a control plan

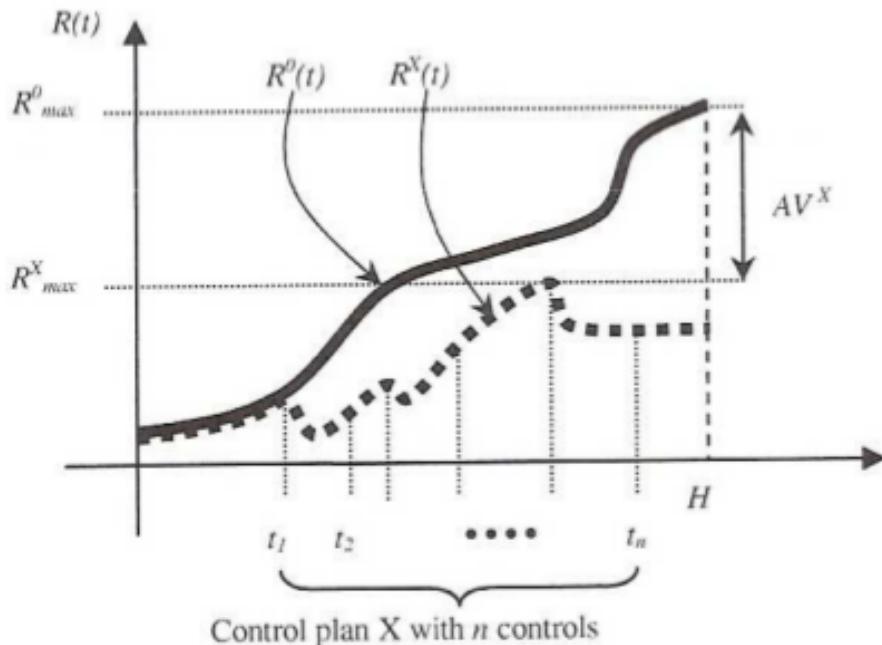


FIGURE 6.1.2. Potential Loss when no control plan

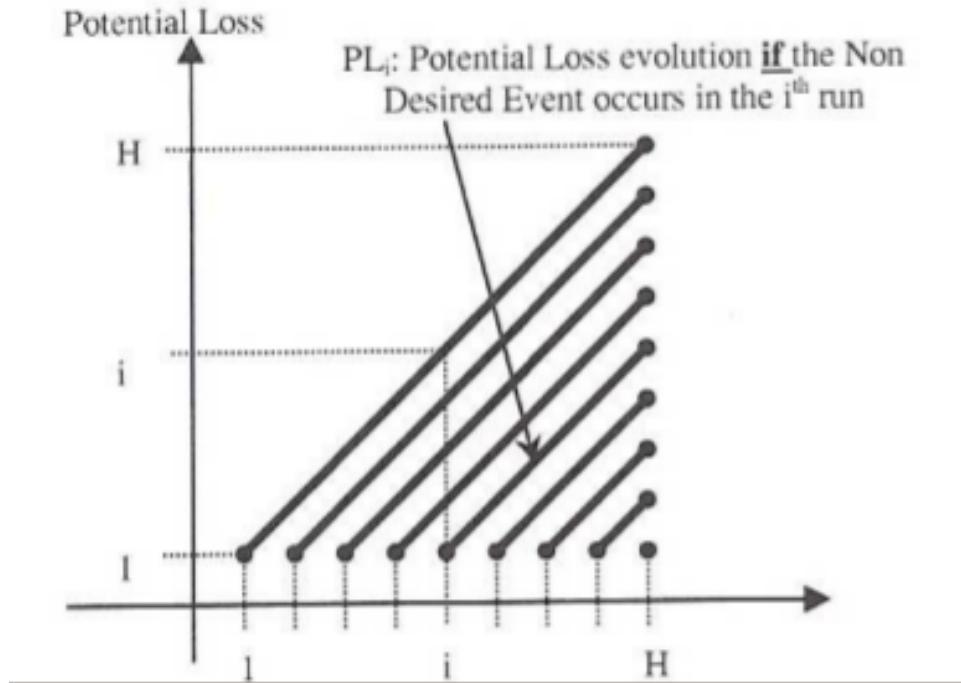
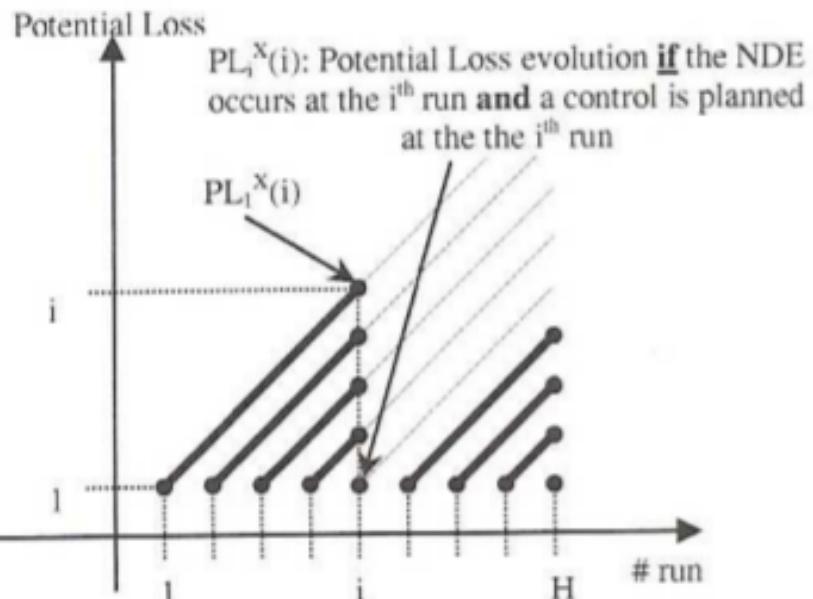


FIGURE 6.1.3. Potential Loss for a control with one control

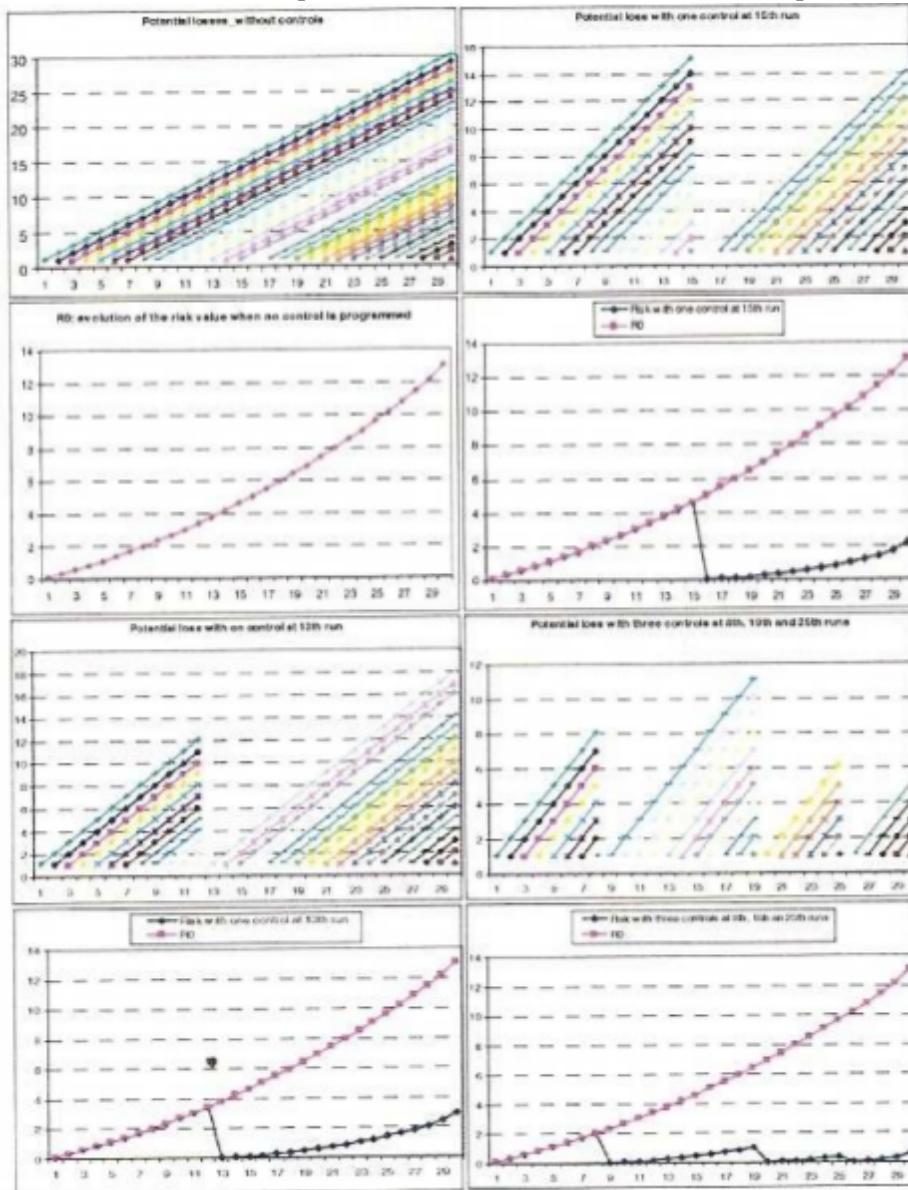


Lorsque les contrôles (les mesures et les actions correctives en cas de détection) sont prévues, la perte potentielle est différente parce que :

- Si le résultat de la mesure envisagée est "OK", il n'est pas nécessaire d'envisager la perte potentielle qui correspond à l'occurrence de l'évènement non-désiré précédant la mesure (cf. Fig 6.1.3).

**CHAPTER 6. AN APROACH FOR OPERATIONAL RISK EVALUATION AND ITS LINK TO CONTROL PLAN**

FIGURE 6.2.1. Examples of risk evolution for different control plans



- Si la mesure n'est pas "OK", une action imminente qui va "réinitialiser ou modifier" la probabilité de l'événement non-désiré, par conséquent la perte potentielle devient "une perte probable" ou "perte prouvée".

En supposant que la loi de distribution de la probabilité des événements non-désirés sur l'horizon considéré (H) est connue, la fonction de l'évolution du risque avec un plan de contrôle X pourrait être calculée comme suit :

$$R^x(i) = \sum_{j=1}^i (i - j + 1).PL_j^i(i).Pr_j$$

## 6.2. Résultats

L'évolution du risque dépend des plans de contrôle, en fonction de leurs nombre de contrôles ainsi que de leurs répartitions sur l'horizon (cf. Fig 6.2.1) :

CHAPTER 6. AN APROACH FOR OPERATIONAL RISK EVALUATION AND  
Rapport Final ITS LINK TO CONTROL PLAN

L'objectif futur de cette approche consiste à optimiser les plans de contrôle en fonction des exigences de l'entreprise.

**6.3. Remarques**

Cette approche nous semble déjà été mise en place chez STMicroelectronics mais cette dernière ne permet pas de satisfaire les exigences de l'industriel. Les plans de contrôle (tâche de qualité) exigent à chaque fois un temps d'improductivité, or d'après l'étude plus il y aura de contrôle plus le risque est réduit, donc cette approche est industriellement non faisable.

## CHAPITRE 7

# Optimizing Return on inspection through defectivity smart sampling

Avec la croissance de la complexité et du besoin de la réactivité des processus, la méthode d'échantillonnage est devenue limitée face aux exigences. Pour combler le défaut de cette méthode, nous avons remarqué que le contrôle de certains échantillons peut être éviter sans perte d'information notable sur la série de produits. Un algorithme[6] a été développé pour identifier et contrôler ce critère sur les produits inspectés inutilement.

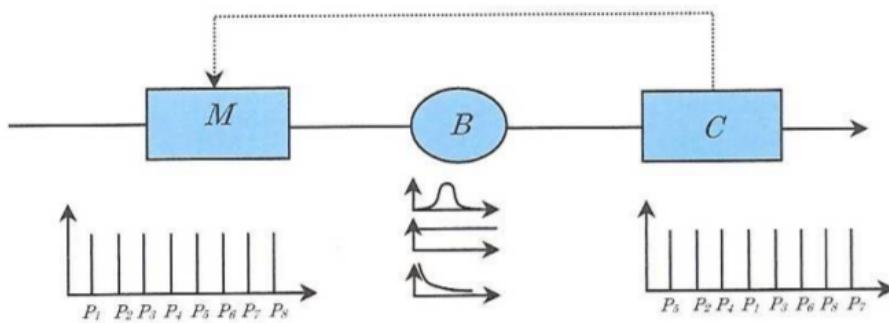
### 7.1. Description de l'algorithme

Sur un cas simple, nous allons prendre en compte le contrôle de la contamination d'une machine de fabrication T (outil de gravure). Le risque de contamination, D, augmente avec le nombre de plaques produites. Si le contrôle est marqué comme "valide", l'outil de traitement et tous les produits traités entre le dernier contrôle "valide" et l'actuel sont validés. Sinon, les actions sont déclenchées sur la machine T et sur la liste des lots potentiellement "non-valide". Les différents retards entre l'outil et le dispositif de contrôle de processus suivent une loi stochastique comme l'illustre la figure 7.1.1. Dans l'étude de cas, l'analyse des données historiques révèle que ce temps d'attente suit une loi gamma.

Le dynamisme du processus est illustré sur la figure 7.1.2 : lot P1 est traité en premier et contrôlé en quatrième en raison du tampon. Certains lots peuvent être contrôlés avant d'autres qui ont été traités plus tôt. Toutefois, le risque de contamination est un phénomène croissant et monotone, nous supposons que si le lot P4 a été validé alors le lot P1 l'est également et il n'est pas nécessaire de le contrôler.

La décision de maintenir ou de ignorer l'inspection d'un lot de produits est basée sur la valeur de sa réduction des risques. Cet indicateur est susceptible de changer à chaque fois qu'un autre lot est inspecté. La figure 7.1.2 illustre cette évolution. Cet indicateur est ensuite utilisé pour trier la pile de lots à inspecter, à

FIGURE 7.1.1. The production system



CHAPTER 7. OPTIMIZING RETURN ON INSPECTION TROUGH  
Rapport Final DEFECTIVITY SMART SAMPLING

FIGURE 7.1.2. Risk reduction variation

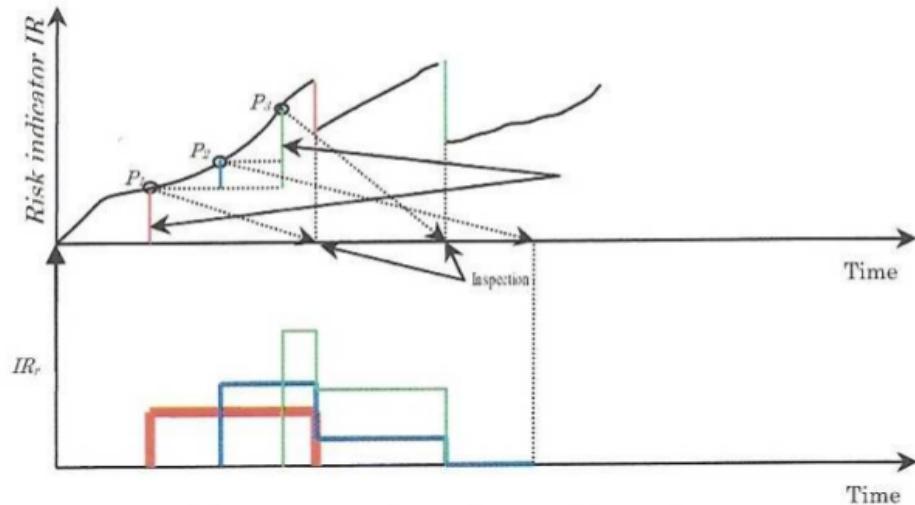
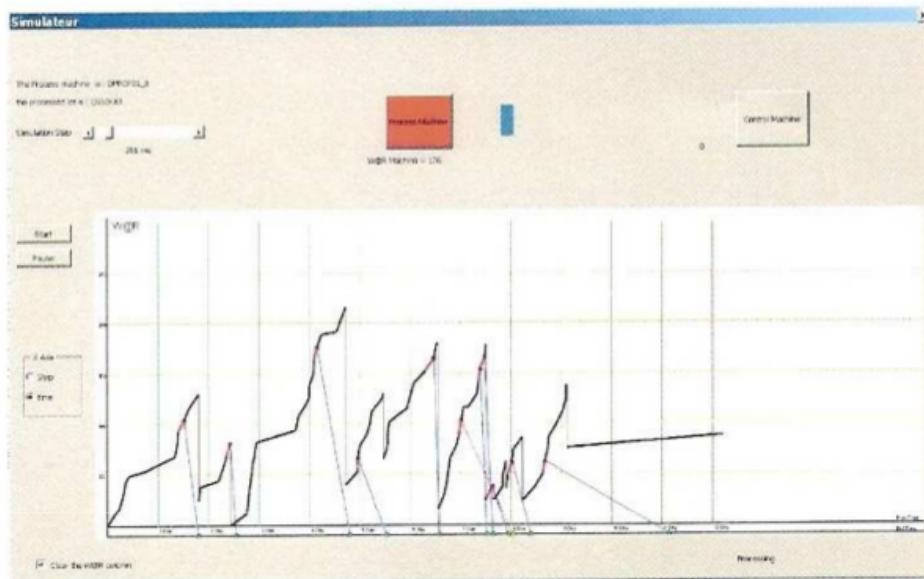


FIGURE 7.1.3. Wafer-At-Risk prototype

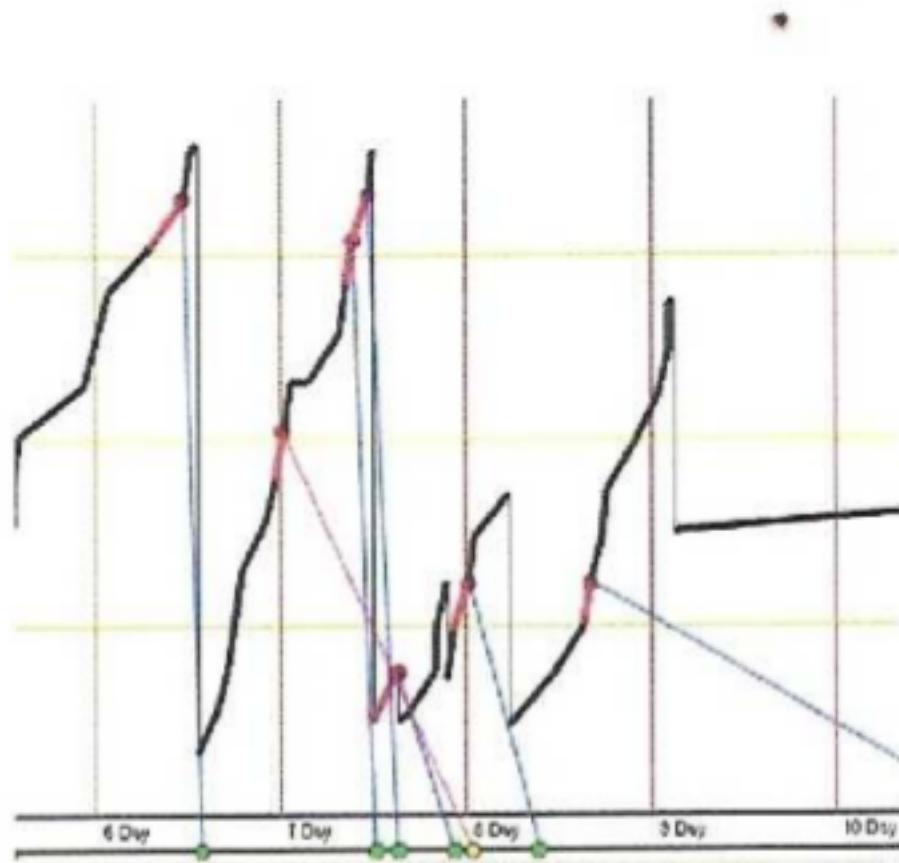


partir des lots qui induisent les plus fortes réductions de risque, l'inspection de ces lots peut être alors ignorée.

Un démonstrateur a été construit pour évaluer l'évolution du Wafer-At-Risk basée sur des données historiques (cf. Fig 7.1.3).

Un cas "ignoré" est illustré à la figure 7.1.4. Un prototype en temps quasi-réel (en exécution toutes les 30 minutes) a été mis en œuvre afin d'aider l'opérateur à sélectionner les lots qui seront inspectés permettant de réduire l'évolution des risques.

FIGURE 7.1.4. Zoom on case of skip



## 7.2. Résultats

L'algorithme a été exécuté sur plus de trois mois de production. Nous avons observé que plus de 35% des plaques inspectées peuvent être ignorées, parce qu'elles n'apportaient aucune information supplémentaire.

## 7.3. Remarques

Cet algorithme est intéressant mais il n'est pas suffisant pour notre problème. Ceci dit, il pourra être implémenté en complément avec une autre solution plus appropriée à notre cas d'étude.

## CHAPITRE 8

# Utilisation du jugement d'expert

Le jugement d'expert[?] présente différents éléments qui peuvent être intéressants dans le contexte de ce PFE, nous dénoterons une affiliation avec l'Analyse des Modes de Défaillances de leurs Effets et de leurs Criticité, méthode communément appelée AMDEC. Nous citerons également des éléments sur l'utilisation des avis d'experts.

### 8.1. Utilisation de l'avis d'experts

En sûreté de fonctionnement, comme dans d'autres domaines, le jugement d'expert peut être considéré comme une des réponses possibles à un problème technique de retour d'expérience ou d'aide à la décision. Il peut compléter ou suppléer les données objectives, lorsque ces données sont peu nombreuses ou inadaptées. Il est souvent la seule source d'information disponible dans un processus décisionnel. Cet ouvrage, focalisé sur les besoins en fiabilité, montre la démarche à suivre pour prendre en compte l'expertise. Cette démarche est illustrée par plusieurs applications réelles concernant principalement l'estimation de durée de vie, l'optimisation de la maintenance, l'aide à la conception et l'aide à la décision. Les différentes étapes du recueil et du traitement de l'expertise y sont décrites : la sélection des experts, l'élaboration du questionnaire à poser aux experts, l'information préalable de l'expert, l'estimation et l'agrégation des expertises, le traitement statistique de l'expertise.

### 8.2. AMDEC

AMDEC (l'acronyme d'Analyse des modes de défaillance, de leurs effets et de leur criticité) est une démarche déductive qui consiste à identifier au niveau d'un système donné, les modes potentiels de défaillance de ses éléments, leurs causes et leurs effets.

Une fois l'état d'art du système est réalisé nous essaierons d'identifier de manière systématique les modes de défaillance potentiels, ensuite nous identifierons pour chaque mode de défaillance les causes qui l'ont provoqué, nous lui associerons un indice de fréquence qui permet de voir combien de fois nous pourrons apercevoir cette même défaillance autrement dit l'indice de fréquence, ses effets, son indice de gravité (classe de sévérité), les mesures mises en place pour détecter la défaillance et son indice de détection (classe de probabilité de détection).

### 8.3. Démarche

La démarche est simple, elle est basée sur une analyse complète du système qui nous permettra d'identifier les fonctions offertes par le système ainsi que les contraintes d'utilisation et d'environnement, les paramètres critiques à mettre sous contrôle et sur lesquels les analyses type AMDEC porteront. Ainsi le périmètre sur lequel l'AMDEC doit être réalisée sera identifié. Les étapes de la démarche sont :

- Décomposition fonctionnelle
- Identification des modes de défaillance
- Classement de chaque mode de défaillance
- Synthèse des résultats
- Identification des méthodes de détection des défaillance et des actions préventives ou correctives

la figure 8.3.1 présente un tableau récapitulatif de toute la démarche

#### 8.4. Objectif de démarche

Maîtriser la démarche AMDEC pour pouvoir :

- Prévoir et réduire les défaillances potentielles et leurs effets
- Améliorer le fonctionnement des processus
- Optimiser la conception

#### 8.5. Avantages

Evaluer et garantir les 4 propriétés de la sûreté de fonctionnement qui sont :

- La fiabilité
  - La maintenabilité
  - La disponibilité
  - La sécurité
- Obtenir le coût maximum de rendement globale des équipements de production.
- Réduire les coûts de maintenance préventive
- Réduire les coûts de maintenance corrective.

#### 8.6. Inconvénient

Elle ne permet pas de prendre en compte la combinaison de plusieurs défaillances en même temps. Et son utilisation peut devenir fastidieuse pour les processus complexes.



Sous-système : moteur-compresseur	Analyse des modes de défaillance de leurs effets et de leurs criticités						Action à engager
	élément	Mode de défaillance	cause	effet	détection	Criticité	
						F G D C	
bâti	Renfermer le compresseur	Ovalisation d'alésage portant les roulements	usure	Détériorations des paliers	-Bruit -échauffement	1 2 2 4	-Réalisation des chemises pour les paliers. -changement du bâti
vilebrequin	Transformer le mouvement de rotation	Usure au niveau des paliers lisses	frottements	Mauvais fonctionnement de compresseur	bruit	1 2 4 8	-changement des paliers lisses -remplacement du vilebrequin
bieille	Transmettre le mouvement au piston	-cassure -fissure	-fatigue -mauvaise conception	Pas de mouvement	visuel	1 4 2 8	-Changement de la bieille -nouvelle conception
Les écrous	La fixation de la croise et la tige	Fissuration de taraudage	- choc -surcharge	Arrêt du compresseur	visuel	1 4 2 8	Changement de l'écrout
Bague racleuse	Assurer l'étanchéité	usure	fatigue	-échauffement	Fuite d'huile	1 4 2 8	Changement de la bague
glissière	Guidage de piston	usure	frottement	vibration	bruit	2 2 2 8	-vérifier le système de graissage -changement de la glissière
Crosse de piston	Orienter le mouvement de translation	usure	frottement	vibration	bruit	2 2 2 8	-vérifier le système de graissage
Tige de piston	Assurer le déplacement de piston	-criques -flambage	-corrosion -surcharge	Mauvais fonctionnement	-bruit -échauffement	2 2 3 12	-Remplacement de la tige -traitement de la tige
chemise	-Renfermer le piston -chambre de compression	usure	frottements	Mauvaise compression	-faible débit au refoulement	3 2 2 12	Remplacer la chemise
cylindre	Renfermer le compresseur	déformation	chocs	Mauvais fonctionnement	visuel	1 2 1 2	Changer le cylindre

FIGURE 8.3.1. Tableau récapitulatif de la démarche AMDEC

## CHAPITRE 9

# Optimized Design of Control Plans Based on Risk Exposure and Resources Capabilities

### 9.1. Introduction

Dans les grandes lignes de fabrication, la plupart des débris importants viennent de la croissance ce qui influence l'incertitude de la santé des produits. Même si le processus est conçu de telle façon qu'il protège le système de fabrication contre des événements pareils, le contrôle d'incertitude est à peine efficace du point de vue opérationnel.

De nombreuses sources de données et de techniques appliquées aujourd'hui sont nécessaires pour maîtriser le processus au niveau de l'Angstrom<sup>1</sup>. Ainsi qu'une connexion entre la production et le contrôle de la qualité tourne la gestion de matériels à risque dans des tâches compliquées, sachant que le matériel est lié à la compétitivité des usines il va falloir renforcer l'approche de contrôle de processus.

L'approche standard utilisée pour exprimer le risque dans la fabrication des semi-conducteurs est l'AMDEC<sup>2</sup>, dans la plupart des cas les statistiques et les plans de contrôle sont gérés par la limite de capacité ou par la productivité / temps du cycle, nous l'appellerons la réduction des étapes de la « non valeur ajoutée ».

Le plan de contrôle est chargé dans MES<sup>3</sup> par l'échantillonnage des règles qu'il soit défini ou pas dans l'AMDEC . Ces règles sont basées sur :

- La fréquence du processus
- Les événements
- Le produit ou un lot de caractéristiques
- Quelques exceptions

L'approche traditionnelle proposée dans la plupart des MES existants est très limitée, par ailleurs un ensemble d'alternatives a été proposé (dans une conférence de la fabrication avancée des semi-conducteurs « Adaptive Metrology Sampling techniques enabling higher precision in variability detection and control »).

### 9.2. Description de l'approche

Dans cet article, les limitations traditionnelles que l'approche possède quand nous l'appliquons à une fabrication avancée avec un nombre de produits variés sont passées en revue, discutées et leurs facteurs principaux en liaison sont présentés.

Dans le but de donner une méthode cohérente pour le planning du contrôle de la qualité atteignant ses limitations, l'approche proposée dans la Fig9.2.1 consiste en deux étapes :

---

1.  $10^{-10}$  mètre

2. L'analyse des modes de défaillance, de leurs effets et de leur criticité

3. Manufacturing Execution System

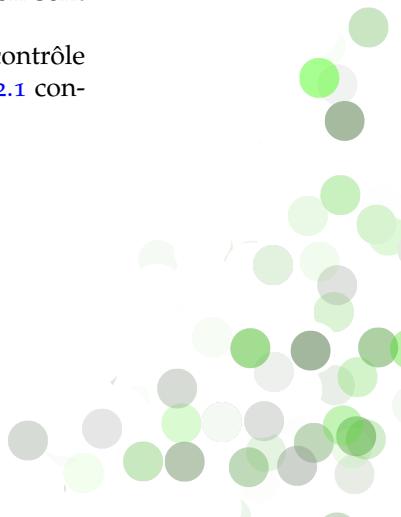
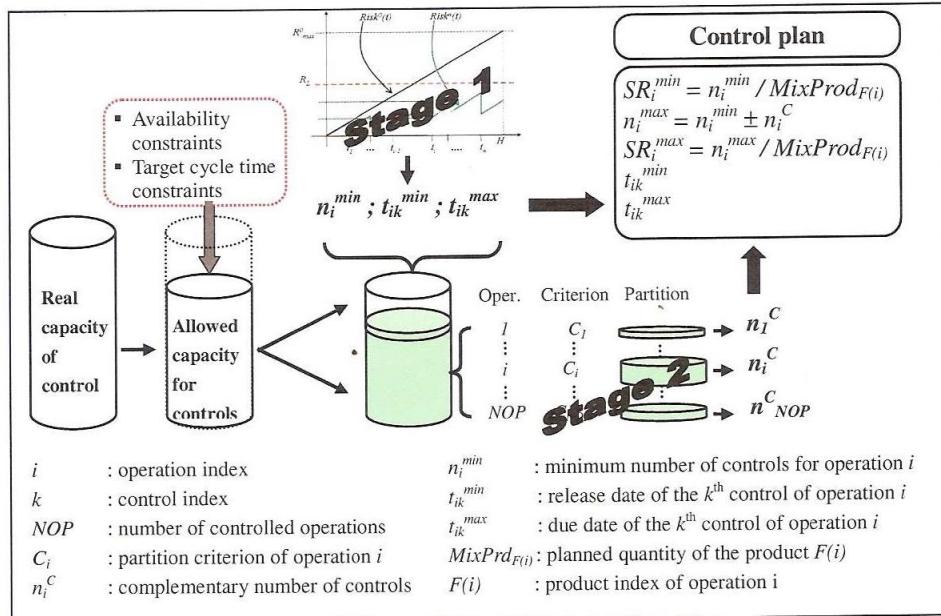


FIGURE 9.2.1. Approche proposée



- (1) utiliser une répartition des risques pour définir un plan de contrôle minimal qui assure un certain niveau de risque pendant l'horizon considéré.
- (2) ajuster le plan de contrôle raffiné dans la phase 1 en divisant le reste (ou le manque) de la capacité selon un critère lié au processus et aux capacités de métrologie. Dans cette phase nous prenons aussi en considération les disponibilités et les contraintes des capacités des ressources de contrôle.

En plus d'ôter les barrières entre le plan de contrôle et L'AMDEC, nous démontrons que la nouvelle approche du risque permet d'obtenir un plan de contrôle optimisé qui garantit une utilisation justifiée d'inspection disponible ou des capacités de contrôle en limitant l'exposition aux différents risques.

### 9.3. Remarque

Cet article propose une nouvelle méthode pour élaborer un plan de contrôle en se basant sur 2 étapes :

**La phase 1:** consiste à identifier les risques possibles et faire une première répartition de risque, dans le but de construire un plan de contrôle par la suite.

**La phase 2:** consiste à ajuster le plan construit dans la phase 1 en ajoutant ou enlevant des contrôles selon la capacité de contrôle possible.

Pour utiliser cette méthode il faut identifier tous les risques possibles ainsi que la capacité de contrôle.

## CHAPITRE 10

# Computation of wafer-At-Risk from theory to real life demonstration

La fabrication de semi-conducteurs devient de plus en plus complexe et le contrôle exhaustif des plaques n'est pas réalisable d'un point de vue financier. La méthode d'échantillonnage se révélait limitée et insuffisante par rapport aux exigences. L'objectif de cette étude consiste donc à évaluer le risque et optimiser l'utilisation des équipements de contrôle.

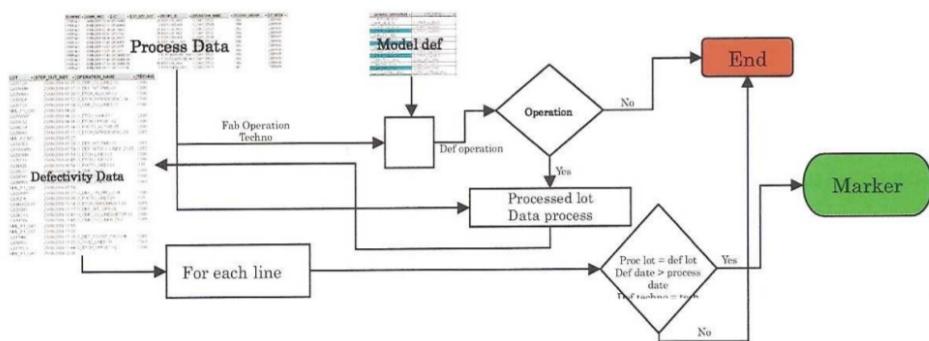
L'évaluation des risques de chaque outil est caractérisée par le Wafer-At-Risk[7]. L'information apportée par chaque lot, qui peut être inspecté, est appelée Wafer-At-Risk-Reduction. Ces deux informations ne sont pas exploitables directement, d'où la nécessité de développer un algorithme pour les calculer.

### 10.1. Description de l'algorithme

La première étape consiste à identifier les lots qui ont été effectivement inspectés. La première difficulté est d'identifier le Defectivity Work Requests (DWR). DWR sont des demandes spéciales issues de l'ingénierie qui met le lot en attente pour une inspection en défectivité plus approfondie. Les archives des mesures sont utilisées et combinées avec les données des processus pour identifier et marquer tous les lots inspectés (cf. Fig 10.1.1).

Après cela, la valeur du Wafer-At-Risk est calculée pour chaque outil du processus. Cette valeur est augmentée avec la quantité des lots de plaques à chaque événement du processus. Cette valeur est diminuée grâce au Wafer-At-Risk-Reduction quand le lot est inspecté. En vue de la quantité de données et la complexité des relations à prendre en compte, l'horizon de traitement a été réduit à une semaine. Afin de faire correspondre les données entre les semaines successives, les conditions aux limites ont été définies, qui se composaient de la liste des lots ayant été traitées après le dernier lot inspecté (cf. Fig 10.1.2).

FIGURE 10.1.1. Flag algorithm



**CHAPTER 10. COMPUTATION OF WAFER-AT-RISK FROM THEORY TO  
Rapport Final**  
**REAL LIFE DEMONSTRATION**

FIGURE 10.1.2. Boundary conditions

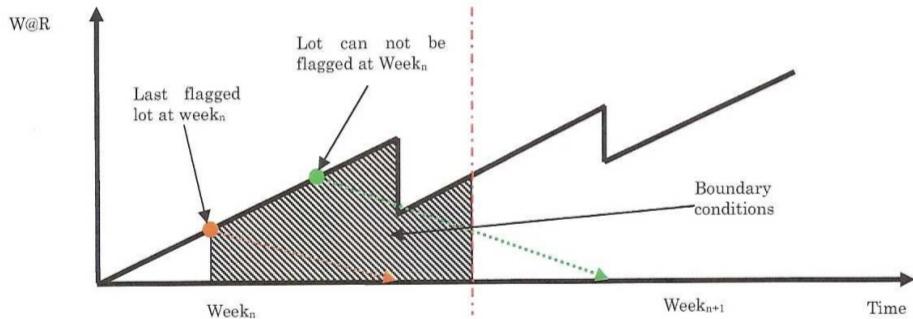
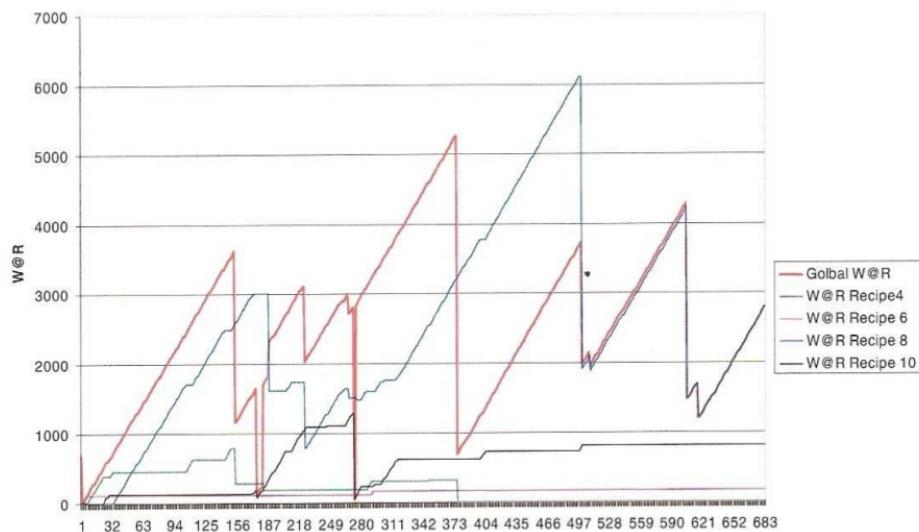


FIGURE 10.1.3. Global W@R and W@R by recipe



Une autre amélioration de l'algorithme a été la définition du Wafer-At-Risk par la recette, du type de recette, de la technologie ou du module d'outils. Le modèle de contrôle de défectivité devait être affiné pour tenir compte de ces paramètres et l'algorithme a ensuite été ajusté pour calculer les différents types de Wafer-At-Risk (cf. Fig 10.1.3).

## 10.2. Résultats

Les résultats obtenus ont été couplés avec un autre algorithme permettant de réduire jusqu'à 35% des inspections inutiles. Un contrôle peut être évité si le Wafer-At-Risk-Reduction du lot marqué est inférieur ou égal à zéro.

De plus l'algorithme a été amélioré pour être plus réactif sur la ligne de production, son implémentation peut être ramenée à une période de 30 minutes.

## 10.3. Remarques

Les détails de cette approche ne sont pas suffisants pour conclure sur sa faisabilité pour notre cas d'étude. Comment peut-on s'en passer de l'historique afin de faire du quasi temps réel ? Le résultat semble être pertinent que pour une

CHAPTER 10. COMPUTATION OF WAFER-AT-RISK FROM THEORY TO  
Rapport Final

REAL LIFE DEMONSTRATION

certaine recette spécifique. Supposons que cet algorithme est applicable, cette méthode pourrait donc être l'une des solutions de notre problème.



## CHAPITRE 11

# Analyse du risque

## 11.1. Méthode générale

L'analyse du risque En trois étapes (Cf Fig11.1.1) :

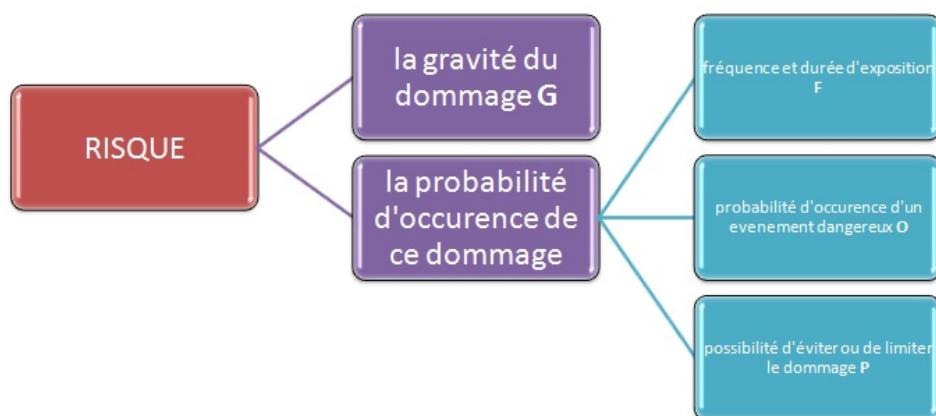
- Détermination des limites de la machine : il faut fixer les signaux de l'appréciation du risque, ensuite il faudra décrire et prendre en compte l'environnement dans lequel est utilisé la machine et les facteurs qui peuvent avoir une influence sur cette dernière.
- Repérage (identification) des phénomènes dangereux. Dans notre projet les plaques de silicium une fois exposées à un phénomène dangereux peuvent être affectées et subir des dommages. Le repérage des phénomènes dangereux est une étape longue et demande beaucoup de précision.
- Estimation du risque : l'estimation du risque consiste à comparer entre elles les différentes situations dangereuses constatées. Le risque peut être défini comme une composition entre la gravité du dommage (notée G) subit par la plaque de silicium et la probabilité d'occurrence de ce dommage. L'occurrence d'un dommage est décomposée en 3 parties :
  - (1) La fréquence et la durée d'exposition aux phénomènes dangereux (notée F)
  - (2) La probabilité d'occurrence d'un évènement dangereux (notée O)
  - (3) La possibilité d'éviter ou de limiter le dommage (notée P).

## 11.2. Gravité du dommage (G)

La gravité du dommage peut être estimée en prenant en compte le taux de rejet de la plaque de silicium. On a deux possibilités :

**G1:** léger défaut (normalement réversible).

FIGURE 11.1.1. Les éléments du risque



**G2:** grave défaut (normalement irréversible, y compris la destruction de la plaque).

### 11.3. Fréquence ou durée d'exposition au phénomène dangereux (F)

L'exposition peut être estimée en prenant en compte :

- le besoin d'accéder à la zone dangereuse ;
- la raison de l'accès ;
- le temps passé dans la zone dangereuse ;
- la fréquence d'accès.

deux choix se présentent :

**F1:** De rare à assez fréquente ou courte durée d'exposition ;

**F2:** De fréquente à continue ou longue durée d'exposition.

### 11.4. Probabilité d'occurrence de l'événement dangereux (O)

La probabilité d'occurrence d'un événement dangereux peut être estimée en tenant compte :

- des données de fiabilité et d'autres données statistiques
- de l'historique des défaillances
- de l'historique des atteintes des plaques

trois choix proposés :

**O1:** Très faible (de très faible à faible). Technologie stable, éprouvée et reconnue pour les applications de sécurité, robustesse du matériel ;

**O2:** Faible (de faible à moyenne). Événement dangereux lié à une défaillance technique ;

**O3:** Élevée (de moyenne à élevée). Événement dangereux entraîné par une action. Principes généraux de gestion du risque.

### 11.5. Possibilité d'évitement du dommage (P)

La possibilité d'évitement permet d'empêcher que le dommage se produise ou de le limiter, en fonction :

- de la rapidité d'apparition de l'événement dangereux ;
- de la conscience de l'existence du phénomène dangereux ;
- de la possibilité d'éviter ou de limiter le dommage

deux choix proposés :

**P1:** Possible dans certaines conditions ;

**P2:** Impossible ou rarement possible.

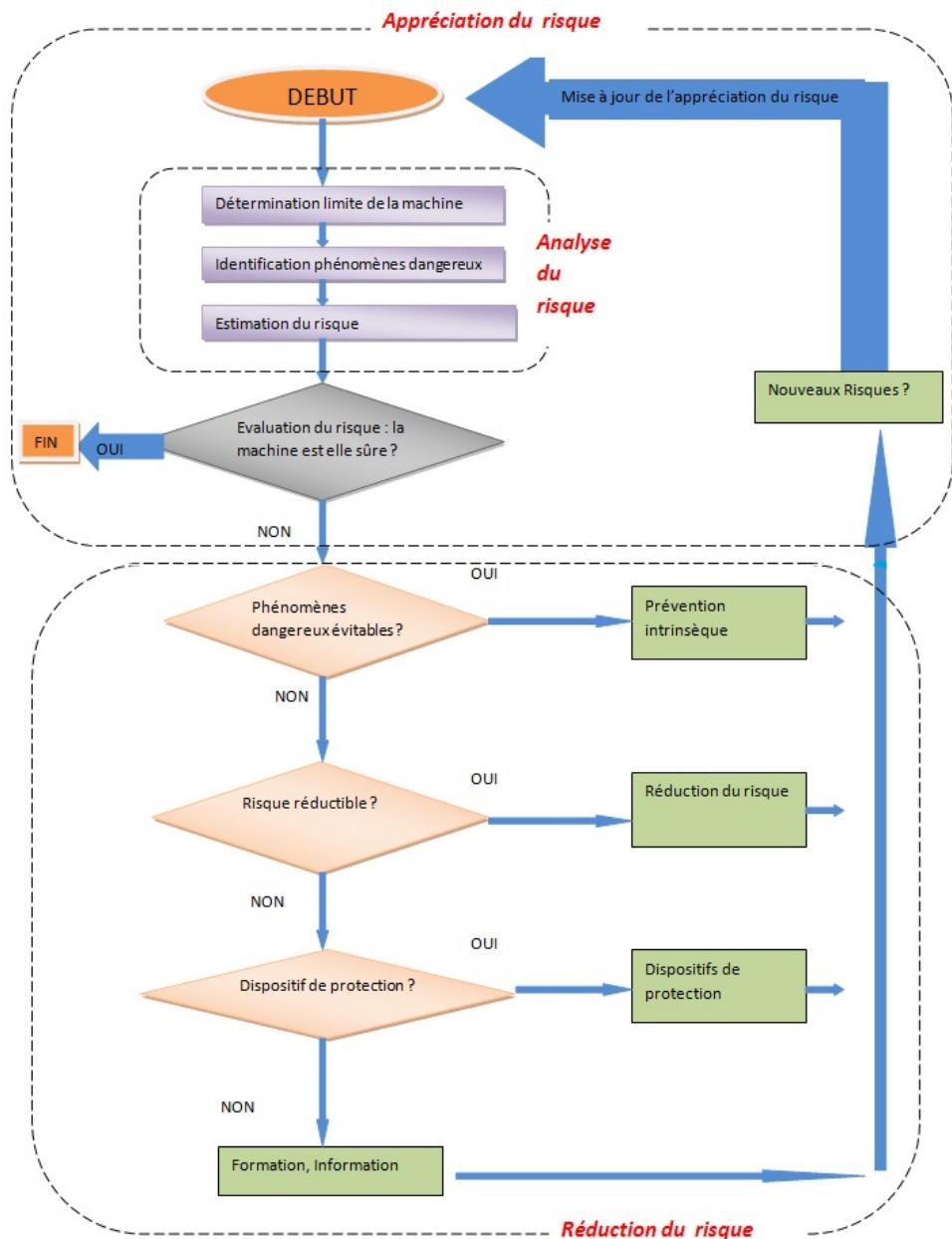
Pour définir l'indice de risque nous utiliserons un logigramme de risque en combinant les résultats des 4 paramètres définis précédemment.

### 11.6. Réduction du risque

Cf Fig 11.6.1.

La figure présente la gestion du risque sous deux parties : l'appréciation du risque précédemment développée et la réduction du risque que nous allons expliqués.

FIGURE 11.6.1. Réduction du risque



### 11.7. Élimination des phénomènes dangereux et réduction du risque :

La prévention intrinsèque peut être définie comme l'élimination du phénomène dangereux et le chemin qui mène vers des situations sécuritaires. Par conséquent la qualité des plaques de silicium va dépendre de la machine utilisée et donc tout se joue lors de la conception de cette machine. Les caractéristiques de cette dernière doivent participer à la réduction du risque.

**11.7.1. Dispositifs de protection :** Une fois la prévention intrinsèque mise en place , les dispositifs de protection viennent prendre place dans le processus , par exemple la mise en place de détecteurs de défaillance après chaque machine.

**11.7.2. Formation et information :** Dans le cas où il y a des phénomènes dangereux persistant ou qui ne peuvent pas être éliminés , une formation sera donnée aux employés sur les différents mode de défaillance mais aussi sur le mode d'utilisation des machines de telles sortes qu'ils puissent détecter quand cette dernière commence à produire des pièces défectueuses.



## CHAPITRE 12

# Bilan sur l'état de l'art

Durant notre étude de l'art nous avons pu traiter différents documents qui se rapproche plus ou moins de notre sujet mais qui avaient toutes un rapport avec ce dernier. Dans ce qui suit nous allons vous présenter les résultats de l'étude de l'art ainsi que les motivations pour retenir ou non une méthode et par la suite l'approfondir et l'appliquer à notre étude :

### 12.1. La démarche AMDEC

Une démarche [?] qualitative et quantitative qui se traduit par l'analyse des modes de défaillance de leurs effets et de leurs criticités.

- Elle repose sur une décomposition fonctionnelle de la machine, une identification des modes de défaillances afin de les classés ainsi que des méthodes de détection des défaillances et des actions préventives ou correctives.
  - L'avantage de cette démarche est d'évaluer et de garantir : la fiabilité, la maintenabilité, la disponibilité, la sécurité, le rendement maximum au moindre cout. Par contre cette démarche ne permet pas de combiner plusieurs défaillances, en effet elle ne peut détecter qu'une défaillance à la fois.

En conclusion nous n'allons pas utiliser cette démarche pour la suite de notre étude car ce type d'étude requiert un niveau de détail beaucoup trop important pour une production de ce type (non linéaire, multi produits, grand nombre d'étape de fabrication).

### 12.2. L'étude de la gestion du risque

Cette étude [?] pour la détection et la réduction du risque rentre dans la prévention des phénomènes dangereux, elle a pour objectif d'apprécier le risque et de le réduire

- Une démarche complète et entière, consiste en une action en amont qui gère et corrige le risque en aval.

Cette étude peut être intéressante car elle a pour finalité de réduire le risque, malgré ceci c'est une démarche très longue et qui exige une connaissance parfaite des machines, donc vu les contraintes de temps que nous avons-nous ne retiendrons pas cette étude pour la suite de notre projet.

### 12.3. Plan de contrôle optimisé en se basant sur le risque

Cet article [3] propose une méthode pour élaborer un plan de contrôle en 2 étapes : d'abord consiste à identifier les risques possibles et de faire une répartition de risque et puis à ajuster le plan construit en ajoutant ou en supprimant des contrôles selon la capacité.

Cette méthode a pour inconvénient de devoir définir au préalable tous les types de risque et la capacité des ressources de contrôle

#### 12.4. An Approach for Operational Risk Evaluation and its Link to Control Plan

Un article [4] qui a pour idée d'évaluer les risques des machines, en effet à chaque traitement la fiabilité du processus diminue (c'est l'usure de la machine) jusqu'à la tâche de qualité (maintenance)

- Son avantage est de réduire fortement le risque du processus. D'autre part cette méthode a pour inconvénient la proportionnalité de la réduction du risque par rapport au nombre de plan de contrôle.

Cette méthode très gourmande en arrêt de la chaîne de production lors des plans de contrôle ne sera pas développée par la suite car elle est industriellement infaisable.

#### 12.5. Optimizing Return On Inspection Through Defectivity Smart Sampling

Évoque une méthode [6] d'échantillonnage "intelligente" permettant d'identifier les produits qui ont besoins de contrôle sans la perte d'information

- Son avantage est de réduire le nombre de contrôle non nécessaire par contre les résultats ne sont pas suffisant par rapport aux exigences industrielles
- Cette méthode ne sera pas retenue même les résultats sont intéressants. En effet cette approche ne permet pas réduire les risques en défectivité mais seulement l'échantillonnage pour la mesure.

#### 12.6. Computation of Wafer-At-Risk from Theory To Real Life Demonstration

Introduit une méthode [7] d'évaluation des risques et d'optimisation de l'utilisation des équipements de contrôle en introduisant les indicateurs W@R et W@R-Réduction

- Son avantage est de réduire fortement le risque presque en temps réel par contre elle est complexe et sa faisabilité reste à prouver.

Malgré des promesses de grandes réduction de risque, nous ne pouvons pas nous engager sur sa fiabilité ni sur son utilisation dans notre étude, donc nous ne l'utiliserons pas.

#### 12.7. Operational risk evaluation and control plan design

**12.7.1. Support de Présentation.** Cette approche d'évaluation des risques est basée sur l'évolution du « Potential Loss » (défectivité potentielle).

- Cette évolution est supposée linéairement croissante avec le nombre d'exécution (traitement par la machine), sans contrôle, sur un horizon donné. Ce risque potentiel est réinitialisé, à une valeur seuil, après chaque tâche de contrôle ou de maintenance.
- L'évolution du « Potential Loss » dépend de la gestion (planification) des tâches de qualité et de maintenance. De plus s'ajoute à cela une fonction d'évaluation de la valeur ajoutée des tâches de contrôles mises en place afin de déterminer la bonne planification de ces tâches pour un résultat optimal du risque.

Cependant ce support de présentation n'apporte pas plus information pour être retenu comme solution de notre problème.

#### 12.8. La méthode $D^3$

Cette méthode [2] est quantitative basée sur l'analyse systématique d'ensemble de produit, en effet elle est décrite sous la forme d'un algorithme qui cherche les produits défectueux en optimisant le nombre de test.

- Cette méthode est programmable et a priori correspond au sujet. Son inconvénient et qu'elle demande beaucoup de tests et donc systématique.

Nous avons décidé de garder cette méthode au début, mais une fois que nous avons commencé à faire la correspondance avec notre étude, nous nous sommes rendu compte que cette dernière demande beaucoup de tests dans le cas ou dans le premier échantillon il n'y a aucun éléments défectueux, de plus nous avons une taille de lot fixe et nous ne pouvons faire des tests simultanés sur plusieurs lots et finalement cette méthode imposera l'arrêt de la production plusieurs fois afin de prélever l'échantillons à tester, la taille de ce dernier peut devenir très grande. Cette méthode s'applique dans un cadres ou la taille de l'échantillon n'est pas fixe et que le contrôle n'en est pas perturbé<sup>1</sup>. Malheureusement les contraintes fortes liées au contrôle des lots nous empêchent d'avoir des contrôles des tailles de lot de plus en plus important.

### 12.9. Conclusion de l'état de l'art

Cette absence de solution concrète dans cette industrie, nous a poussé à développer notre propre méthode de détection.

L'idée étant d'utiliser nos contraintes pour développer notre propre algorithme de calcul, nous détaillerons cet algorithme ainsi que ces résultats dans la partie suivante.

---

1. Dans le cadre d'une analyse épidémiologique par exemple, tester un échantillon de sang d'une personne ou cent personnes ne fait pas de différence dans le cas de la recherche d'un virus dans une population

## **Troisième partie**

# **Développement de la solution**



## CHAPITRE 13

# Algorithme du marcheur

### 13.1. Introduction

Au terme de notre étude sur l'état de l'art, nous n'avons pas trouvé de solution adaptable à notre problème, de ce fait il nous est paru nécessaire de développer un algorithme plus adapté au besoin du client.

Par la suite nous allons détailler le modèle que nous avons développé « Le modèle du marcheur » :<sup>1</sup>

Il permet dans un premier temps de déterminer un indice de santé pour les marcheurs (si il est en bonne santé, malade...). Dans un deuxième temps nous nous servirons de cet indice de santé pour déterminer les marcheurs les plus malades, prioriser leur contrôle médical, et remonter plus rapidement aux marcheurs gravement blessés. Afin de représenter ces données nous les avons couplées avec l'avancement du marcheur dans la ville.

L'idée de départ est assez simple, nous proposons de regarder la probabilité d'un marcheur d'être agressé en se baladant dans une ville<sup>2</sup>.

### 13.2. Description de l'algorithme

**13.2.1. Idée de l'algorithme.** Afin de rendre le propos le plus clair possible nous allons détailler les hypothèses utilisées par l'algorithme du marcheur.

- Un marcheur augmente sa « chance » d'être blessé au fur et à mesure qu'il se balade dans la ville .
- Le marcheur qui traverse des quartiers dangereux a plus de chance d'être blessé que dans un quartier tranquille.
- Propriétés du quartier dynamique :
  - Le taux de criminalité d'un quartier augmente linéairement. De manière parallèle la sûreté d'un quartier diminue linéairement en fonction du nombre de marcheurs se promenant dans le quartier. En effet plus de gens passent dans un quartier plus cela attire les bandits. Ce qui entraîne une augmentation direct du  $Taux_{Criminalité}$

$$Sureté_{Quartier} = Sureté_{Défaut} - (Taux_{Criminalité} \times Nombre_{Marcheur})$$

- La criminalité augmente en fonction du nombre de marcheurs qui traversent le quartier.
- La criminalité est remise à un niveau seuil quand la police passe.
- La criminalité est remise à un seuil de confiance si la santé du marcheur est bonne en sortant du quartier. Un médecin contrôle le marcheur, si il est en bonne santé nous en déduisons que la criminalité du quartier a été surévaluée, nous ré-augmentons cette sûreté à un niveau seuil.

---

1. Le nom « modèle du marcheur » est un clin d'œil à la « marche de l'ivrogne » qui est un modèle utilisé dans la thermodynamique et le mouvement brownien.

2. Notre vie à Marseille dans les quartiers nord nous a fortement inspiré pour l'élaboration de cet algorithme

- Le contrôle de la santé d'un marcheur donne de l'information uniquement sur le dernier quartier vu par le marcheur.
- Il y a trois types de marcheurs : un petit, un moyen et un grand
  - Plus un marcheur est petit, plus il aura de chance d'être agressé, en effet les bandits préfèrent prendre le moins de risque possible.
- Nous formalisons le problème :

$$(13.2.1) \quad Indice_{Santé} = Indice_{SantéDéfaut} \times \prod_{i=1}^N (Sureté_{Quartier[i]})$$

- $Indice_{Santé}$  est un indice de confiance plus cette valeur est basse plus un marcheur a de chance d'être blessé et un contrôle du médecin s'impose au plus vite.

**13.2.2. Adaptation de l'algorithme au problème industriel.** Pour adapter notre algorithme à notre problème industriel nous faisons les associations suivantes :

- Un marcheur est un lot (*Batch*)
  - *Marcheur* → *Lot*
- Un quartier est une machine de production (*Workstation*) et la sûreté du quartier sera la fiabilité de la machine
  - $Sureté_{Quartier} \rightarrow F_{Machine}$
- Une descente de police est une tâche qualité sur une machine (*MSE*)
- Un contrôle du docteur est une mesure sur lot (*MSL*)
- Le type du marcheur petit, moyen ou grand
  - $TypeMarcheur \rightarrow Technologie$ , (exemple : mémoire, logique ou mixte).
- L'état de santé du marcheur sera l'indice de santé du lot (ou fiabilité du lot)
  - $Indice_{SantéMarcheur} \rightarrow Indice_{SantéLot}$

### 13.3. L'algorithme

Nous avons défini différents niveaux de complexité à notre algorithme, pour chaque niveau de complexité nous allons détailler les différences. Il est à noter que tous ces calculs sont donnés pour l' $Indice_{Santé}$  d'un seul lot. Nous avons programmer cet algorithme dans une boucle pour comparer les différents lots entre eux. Ensuite nous avons déterminé un classement des lots les plus risqués en fonction de leurs avancements. Cela va nous permettre de savoir rapidement quel produit contrôler en priorité.

**13.3.1. Niveau 0 : Modèle simple.** Nous supposons que les machines ont une fiabilité initiale  $F_{Machine} = 0,5 = 50\%$ , il y a  $N$  étapes de fabrication, et  $Indice_{Santé}$  est la probabilité du lot d'avoir des produits défectueux.

- Pour  $i$ , de 1 à  $N$ 
  - $Indice_{SantéLot} = Indice_{SantéLot} \times F_{Machine[i]}$
- Fin Pour

$F_{Machine[i]}$  est la fiabilité de la machine  $i$ .

**13.3.2. Niveau 1 : Modèle des quartiers.** Nous supposons maintenant que la fiabilité est différente sur chaque machine  $F_{Machine[i]}$

- $Indice_{SantéLot} = Indice_{SantéLotDefaut}$
- Pour  $i$ , de 1 à  $N$ 
  - $Indice_{SantéLot} = Indice_{SantéLot} \times F_{Machine[i]}$
- Fin Pour

Exemple de loi de distribution de  $F_{Machine[i]}$  : Répartition Uniforme, Loi Normale, Loi de Poisson, Loi définie par morceaux (retour d'expérience).

**13.3.3. Niveau 2 : Modèle des quartiers avec criminalité croissante.** Nous supposons maintenant que la fiabilité est différente sur chaque machine et décroît linéairement :

$$(13.3.1) \quad F_{Machine[i]} = F_{MachineDefault} - (\lambda_{Machine} \times N_{Produit})$$

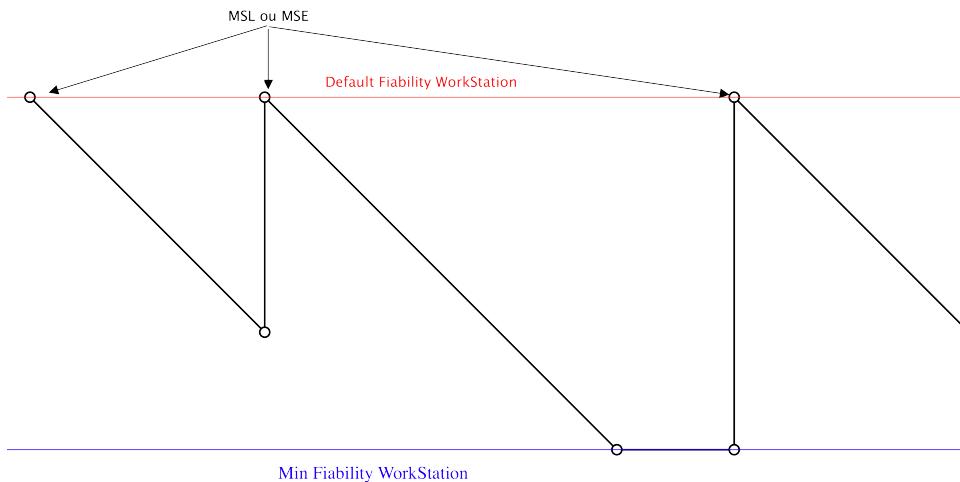
- $N_{Produit}$  : le nombre de produits passés sur la machine depuis le dernier contrôle.
- $\lambda_{Machine}$  : le taux de détérioration de la machine .
- $F_{MachineDefault}$  : la fiabilité maximal par défaut, suite à un contrôle de la machine.

Afin de définir un seuil minimal de Fiabilité de la machine nous avons défini la Fiabilité de la manière suivante :

$$(13.3.2) \quad F_{Machine} = \max(F_{MachineMin}, F_{Machine})$$

- Pour  $i$ , de 1 à  $N$ 
  - $Indice_{SantéLot} = Indice_{SantéLot} \times F_{Machine[i]}$
- Fin Pour

FIGURE 13.3.1. Modèles d'évolution fiabilité (ou la criminalité) d'une machine (ou quartier)



Le graphique suivant représente l'évolution de la fiabilité d'une machine en fonction du nombre de produits qui passent sur la machine. Certains évènement font remonter le niveau de fiabilité de la machine à un seuil par défaut (trait rouge). De plus la fiabilité est minorée par un seuil minimal de fiabilité (trait bleu). Tant qu'aucun évènement de contrôle (MSL ou MSE) se produit sur la machine la fiabilité reste au même niveau.

**13.3.4. Niveau 3 : Modèle du marcheur in-homogène.** Nous supposons maintenant que le type de marcheur (la technologie), est plus susceptible d'être blessé suivant son type intrinsèque

$$F_{Produit} = \beta_{type} \times F_{Produit}$$

avec  $\beta_{type} \in \{\beta_{petit}; \beta_{moyen}; \beta_{grand}\}$ .

Comme dans la vie réelle un grand aura moins de chance d'être agressé qu'un petit.

Afin de faire le parallèle avec les technologies :

- $\beta_{grand}$  correspond aux technologies de logique, qui sont les plus faciles à mettre en œuvre
- $\beta_{moyen}$  correspond aux technologies de mémoires, qui sont légèrement plus complexes.
- $\beta_{petit}$  correspond aux technologies mixtes<sup>3</sup> qui sont les technologies les plus délicates à mettre en œuvre et ayant de fortes chances de générer de la non qualité.

Le calcul s'effectue de la même manière, ensuite nous introduirons simplement le nouveau coefficient de risque du lot ( $\beta$ ).

Nous déduirons la valeur de l'indice de santé après le calcul précédent de la manière suivante.

$$Indice_{SantéLot} = Indice_{SantéLot} \times \beta$$

Il est à noter que les informations de complexité ne nous ayant pas été communiqué, cette partie de l'algorithme n'a pas été implémenté.

#### 13.4. Spécification du Code

Vu la complexité du projet nous avons décidé de développer notre code en Java, la structure des données et l'algorithme nous incitent à utiliser ce type de langage. De plus comme il s'agit simplement de la validation d'algorithme, ce code n'a pas pour vocation d'être développé ultérieurement. En effet le Java est le langage enseigné dans notre formation et permettra une implication de tous.

**13.4.1. Traitement des données.** Concernant les données utilisées il faut préciser que pour des raisons de temps et de moyens nous avons travaillé sur un ensemble de données tronquées. En effet les informations liées aux technologies ont été supprimées.

**13.4.1.1. Analyse des données.** Afin de tester notre algorithme nous avons récupéré un fichier de données de *STMicroelectronics*, afin de bien comprendre ces données nous allons les analyser.

Nous distinguons différents champs de données :

**taskID (Champs Vide):** Le premier champs vide est le numéro de la tâche, chaque tâche est numérotée de manière unique.

**Date:** Il s'agit de la date à laquelle l'évènement étudié a eu lieu. Il est à noter que le fichier de données dont nous disposons fait l'objet d'un historique d'un mois de production filtré, le temps est donné en secondes.

- Le champs de la date est composé de 14 chiffres, exemple : 20101010162720
- Les 4 premiers chiffres représentent l'année, ici 2010
- Les 2 chiffres suivants représentent le mois, ici 10 donc octobre
- Les 2 chiffres suivants représentent le jour, ici 10
- Les 6 derniers chiffres représentent l'heure, ici 162720 qui donne 16 heure 27 min et 20 secondes

**Lot (batch):** Il s'agit du numéro du lot, impacté par la tâche. Dans le cas où ce champ est vide, cela veut dire que la machine ne traite pas de produit mais que nous effectuons une tâche qualité sur la machine.

**Oper (operation):** Ce champs qualifie le type d'opération effectuée sur le lot

<sup>3</sup>. mémoire et logique

**Nbr (quantity):** Représente le nombre de plaques concernées par un événement de process, c'est-à-dire le nombre de plaques passées dans la machine.

**Event:** Permet de qualifier le type d'évènement. Dans notre exemple nous avons trois types d'évènements différents :

**PRP:** représente une étape de fabrication du produit

**MSL:** indique une mesure effectuée sur un lot

**MSE:** indique une mesure réalisée sur un équipement (tâche de qualité, ou « TQ »)

**Tech (technology):** Représente la technologie, nous disposons de cinq technologies différentes. Sur nos données il y a visiblement 5 types de technologie, cependant dans l'idée de rendre réalisable cette étude de faisabilité. Aucun détail n'est donné sur les technologies.

**Eqpt (workstationID):** Représente le numéro de l'équipement, chaque équipement est représenté par un identifiant unique.

13.4.1.2. *Classe des entités.* Afin d'utiliser les données au mieux, nous allons organiser les données sous forme de la structure suivante détaillée dans la suite. Ces classes font partie du package *entities*. Nous allons organiser les données par produit, machine et ordre. Dans le détail :

**MyDate:** classe qui organise une date, il est à noter que cette structure particulière repose sur la classe native Java qui s'appelle *Date* avec quelques petites adaptations.

**Entier year:**

**Entier month:**

**Entier day:**

**Entier hour:**

**Entier minutes:**

**Entier secondes:**

**Traçabilité (Tracability):** structure de la mémoire des événements, un objet de cette classe représente un événement élémentaire de l'usine de production.

**Charactere Tâche:** numéro de tâche.

**Date date:** Donne la date de chaque tâche.

**Charactere Type:** Précise le type de tâche.

**Machine (WorkStation):** Qualifie la ressource qui usine le procédé

**Collection de traçabilité:** « mémoire » de la ressource, c'est elle qui stocke les tâches se produisant sur la machine

**Charactere Nom:** Le nom de la machine

**Lot (Batch):** Classe qui qualifie un lot

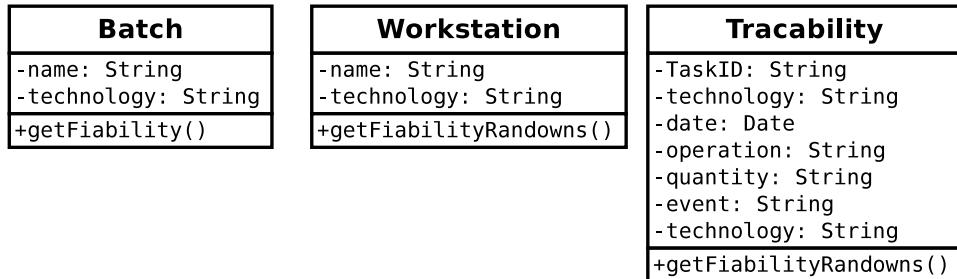
**Collection de traçabilité:** « mémoire » du produit, c'est elle qui stocke les tâches se produisant sur la machine.

**Charactere Nom:** Le nom du produit

**Entier Technologie:** Permet de définir le type de technologie du lot

Cette structure nous permet d'organiser les données de manière logique, et permet de piocher facilement dans les données.

FIGURE 13.4.1. Diagramme de classe des objets entités



Ce diagramme de classe représente les objets sur lesquels nous travaillerons et utiliserons notre algorithme

### 13.5. Implémentation du marcheur

**13.5.1. Marcheur.** Contient l'algorithme qui lance les méthodes de calcul qui se trouvent sur les batchs et workstations. Les méthodes de calcul de fiabilité se font sur l'objet batch et Workstation. En effet à la vue de notre architecture, les données nécessaires aux calculs sont présentes sur ces objets et il est naturel d'associer les méthodes à ces objets.

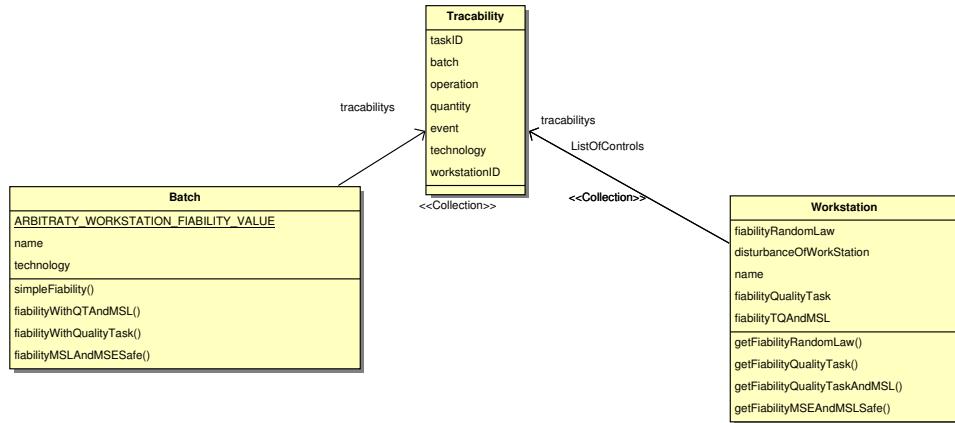
**13.5.2. Lecture de fichier (Storage).** Nous organisons notre lecture de fichier de façon à ranger les informations dans la bonne structure.

Nous définissons une classe qui lit les fichiers et instancie les traçabilités associées à chaque ligne. De plus ce chargeur de fichier est spécifié pour comprendre le fichier donné par STMicroelectronics. Le fichier fourni est une extraction de la base de donnée ST, qui pour des raisons de confidentialité a été anonymisé.

**13.5.3. Extracteur.** Ces classes interprètent les traçabilités extraites par l'extracteur et créent les objets batch, workstation et insèrent les traçabilités associées. Elles font en quelque sorte le tri des données et permettent d'associer chaque traçabilité à la machine et au lot qui lui correspond. De plus ces classes créent les objets et instantient d'autre objet nécessaire à l'exécution de l'algorithme.

**13.5.4. Package entities.** C'est dans ce package que se situe le cœur de l'algorithme. En effet les calculs de fiabilités sont des méthodes associées au objets *Batch* et *Workstation*. Ces objets possèdent comme attributs leur nom, leur technologie et leur mémoire qui est représentée par une collection de *tracabilities* ce qui rend le calcul d'indice de santé plus facile à effectuer (détail du code source en section 15.2 et 15.1).

FIGURE 13.5.1. Diagramme de classe des différentes entités



Ce diagramme de classe représente l'ensemble des classes entities, les Batch et Workstation contiennent des collections de tracability, qui constituent la mémoire de chacun des éléments. De plus

**13.5.5. Classe Constants.** Les connaissances métiers pouvant être à même d'évoluer nous avons effectué une classe constante qui regroupe toutes les données utilisées pour les calculs d'algorithme et peuvent être réglées très facilement. Nous retrouvons dans cette classe les attributs suivants :

**FIELD\_QUANTITY\_IN\_FILE\_ROW:** Il s'agit là d'un paramètre pour lire et interpréter le fichier de départ

**DEFAULT\_VALUE\_WHEN\_EMPTY\_ROW:** Permet de donner le texte à écrire lors de la lecture d'un caractère vide

**DISTURBANCE\_WORSTATION\_PER\_EVENT:** Il s'agit là du dérèglement d'une machine après le passage d'un lot il s'agit de  $\lambda_{Machine}$

**FIABILITY\_DEFAULT\_BATCH:** Il s'agit de l'indice de santé par défaut d'un lot  $Indice_{Santé}$

**FIABILITY\_DEFAULT\_WORKSTATION:** Il s'agit là de la fiabilité d'une machine par défaut  $F_{MachineDefault}$

**FIABILITY\_MIN\_WORSKATION:** Il s'agit du seuil minimal de fiabilité d'une machine  $F_{MachineMin}$

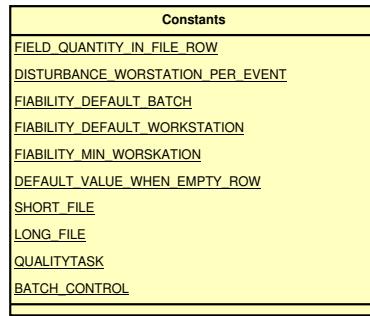
**SHORT\_FILE:** Il s'agit là du chemin des données courtes (pour les tests rapides)

**LONG\_FILE:** Le chemin pour les données complètes

**QUALITYTASK:** La chaîne de caractères qui indique une tâche de qualité sur une machine

**BATCH\_CONTROL:** Chaîne de caractère qui indique un contrôle de lot

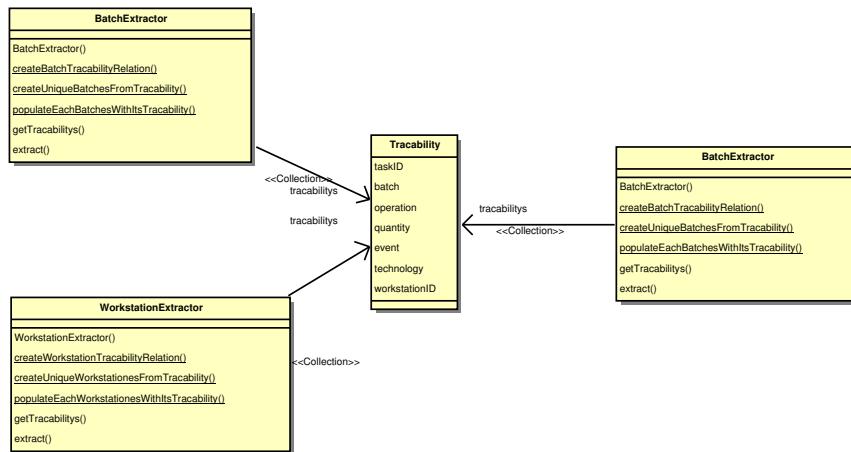
FIGURE 13.5.2. Diagramme de classe des constantes



Cette classe permet de paramétriser l'ensemble du projet en définissant les constantes utilisées dans le projet.

**13.5.6. Package Extractors.** C'est le package qui contient toutes les classes permettant de créer les *Batch* et les *Workstation* à partir de l'ensemble des traçabilités générées par les classes *StoreData*<sup>4</sup>.

FIGURE 13.5.3. Diagramme de classe des extractors



Cet ensemble de classe permet d'affecter les traçabilités au seins des bons lots et des bonnes machines

## 13.6. Résultats

Pour représenter les résultats de l'algorithme, nous utilisons un graphique en deux dimensions dans lequel nous affichons l'indice de santé en fonction de l'avancement du produit (cf Fig 13.6.1, Fig 13.6.2). Ces données ont été obtenues avec les réglages présents réciproquement dans le tableau 13.6.1 et 13.6.2.

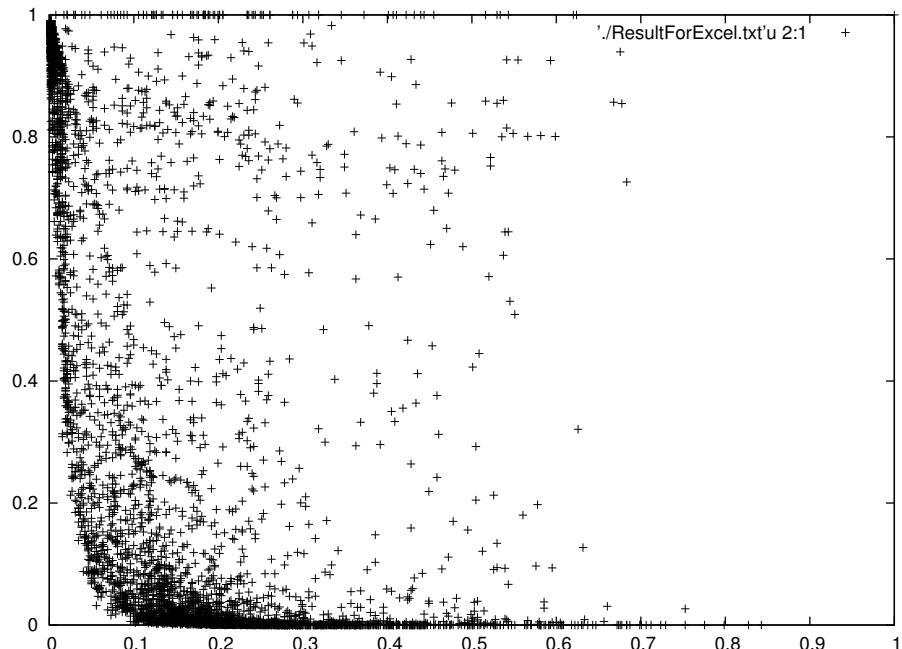
4. Ces classes ne sont pas détaillées, car elles ne font que lire et écrire dans des fichiers. Comme il n'y a rien de nouveau sous le soleil nous passerons cette partie sous silence

TABLE 13.6.1. Données de réglage de l'algorithme

Constante	Valeur
$\lambda_{Dereglement}$	0.0001
$F_{MachineDefaut}$	1
$F_{MachineDefaut}$	0.99
$F_{MachineMin}$	0.1

Valeurs de réglage utilisées pour générer les résultats représentés en figure 13.6.1

FIGURE 13.6.1. Représentation des résultats après calculs



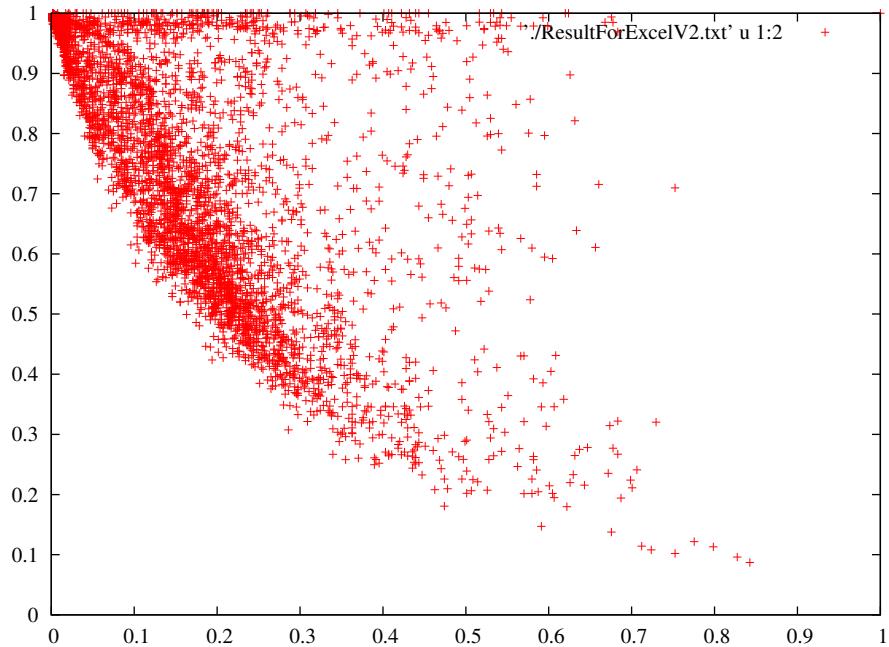
Sur l'axe des abscisses nous avons l'avancement du produit et l'axe des ordonnées représente l'indice de santé. Chaque croix représente un produit à l'instant  $t$ . Les réglages de cette simulation sont donnés dans le tableau 13.6.1

TABLE 13.6.2. Données de réglage de l'algorithme

Constante	Valeur
$\lambda_{Dereglement}$	0.00001
$F_{ProduitDefaut}$	1
$F_{MachineDefaut}$	0.999
$F_{MachineMin}$	0.5

Valeurs de réglage utilisées pour générer les résultats représentés en figure 13.6.2

FIGURE 13.6.2. Représentation des résultats après calculs



Sur l'axe des abscisses nous avons l'avancement du produit et l'axe des ordonnées représente l'indice de santé. Chaque croix représente un produit à l'instant  $t$ . Les réglages de cette simulation sont donnés dans le tableau 13.6.2

### 13.7. Inconvénients de cette méthode

**13.7.1. Complexité.** L'un des plus gros problèmes de ce type de solution est la complexité du code. En effet des boucles imbriquées entre elles font que pour  $n$  données nous avons  $\log(3 \times n)$  calculs. Ce qui est gênant dans ce type de problème car le nombre de donnée est assez considérable et le temps de calcul qui en découle est important.

**13.7.2. Manque de réalisme avec la réalité.** Le code actuel ne fait pas état de la réalité complète. Pour les besoins de l'étude quelques hypothèses simplificatrices ont été faites :

- Les technologies ne sont pas détaillées dans les données traitées.
- Les contrôles MSE et MSL sont sur une technologie particulière et non pas sur la fiabilité globale de la *Workstation* et l'indice de santé du *Batch*. Une adaptation du modèle seraient la bienvenue pour adapter de manière plus fine l'analyse avec le problème. Les *MSL* et *MSE* sont découpées en des sous catégories qui dépendent des technologies analysées.
- Dans notre projet toutes les machines ont le même poids pour le calcul de la fiabilité. Ce qui diffère avec la réalité, c'est que certaines machines ont un poids plus important que d'autres.

Ces propositions constituent des modifications à faire pour un développement ultérieur.

**Quatrième partie**

**Conclusion de l'étude**



## CHAPITRE 14

# Conclusion

Au terme de notre étude nous pouvons revenir sur la mission de notre projet de faisabilité.

Tout d'abord il apparaît qu'aucune méthode simple n'est utilisable facilement pour la détection de la non qualité dans les *chips*.

Au cours de nos recherches bibliographiques deux méthodes ont attiré notre attention : la méthode  $D^3$  et l'approche Wafer-at-risk. Nous avons cherché à adapter ces deux méthodes dans le cadre de notre étude mais sans succès. En effet les contraintes de production et de tests sur les composants nous ont constraint d'abandonner ces deux méthodes. Refusant de rester dans une impasse nous avons décidé de développer notre propre méthode afin de trouver une piste de solution. Pour ce faire nous avons conçu un algorithme que nous avons appelé « l'algorithme du marcheur ».

Pour développer cette solution nous partons du principe que fabriquer un produit de non qualité revient au même principe que de se faire agresser dans une grande ville comme Marseille en se baladant. Le but du programme est d'anticiper grâce aux probabilités et à des méthodes de calculs récursives, avec quelle chance un produit est de non qualité.

Afin de valider notre algorithme nous avons décidé de le programmer. Nous avons donc programmé une maquette en Java qui a permis de sortir des résultats partagés avec STMicroElectronics ainsi qu'une architecture de logiciel. De plus ce développement nous a permis d'envisager les évolutions ultérieurs. En effet il apparaît que notre modèle n'est pas encore assez complexe pour être compatible avec la réalité. Une optimisation du code sera nécessaire pour résoudre le problème des technologies différentes à considérer. Ceci dit nous avons bon espoir que notre méthode fera l'objet d'un re-développement ultérieur et peut-être d'une utilisation dans un cadre industriel. En effet ce type d'application présente un enjeu pour ce type d'industrie.

Sinon d'un point de vue plus général sur le projet, cette expérience nous a permis d'être en contact avec un industriel mais aussi de nous confronter à la réalité du travail d'ingénieur. Gérer, anticiper et comprendre les besoins du client ce qui dans les prochaines semaines seront des compétences que nous déployeront dans un contexte moins protégé que l'école et plus en phase avec la réalité. Ce projet fut donc un bon moyen de tourner une page avant de sauter dans le grand bain du métier d'ingénieur.



## **Cinquième partie**

## **Annexes**



## CHAPITRE 15

# Code Source

## 15.1. Classe Batch

```
/*
 * Copyright (C) 2010 Team-W@R (team-war@prunetwork.fr)
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 */
package fr.prunetwork.teamwar.entities;

import fr.prunetwork.teamwar.Constants;
import fr.prunetwork.teamwar.utilities.MyDate;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Iterator;

/**
 *
 * @author jpierre03+teamwar@prunetwork.fr
 * @author GARCIA Fabien
 * @author NAIT BELKACEM Abdelali
 */
public class Batch {

    private static final double ARBITRATY_WORKSTATION_FIABILITY_VALUE = 0.5;
    private String name;
    private String technology;
    private Collection<Tracability> tracabilitys =
        new ArrayList<Tracability>();

    public Batch(String name, String technology) {
        this.name = name;
```

```
        this.technology = technology;
    }

    /**
     * @return the name
     */
    public String getName() {
        return name;
    }

    /**
     * @return the technology
     */
    public String getTechnology() {
        return technology;
    }

    public void addTracability(Tracability tracability) {
        tracabilitys.add(tracability);
    }

    /**
     * @return the tracabilitys
     */
    public Collection<Tracability> getTracabilitys() {
        return Collections.unmodifiableCollection(tracabilitys);
    }

    public String description() {
        StringBuilder sb = new StringBuilder();
        sb.append(getName());
        sb.append("\t");
        sb.append(getTechnology());
        sb.append("\t");
        sb.append("tracabilitysCount: ");
        sb.append(getTracabilitys().size());
        return sb.toString();
    }

    public double simpleFiability() {
        double fiability = Constants.FIABILITY_DEFAULT_BATCH;
        fiability *= Math.pow(ARBITRARY_WORKSTATION_FIABILITY_VALUE,
            (double) getTracabilitys().size());
        return fiability;
    }

    public double fiabilityWithQTAndMSL(MyDate currentDate) {
        double fiability = Constants.FIABILITY_DEFAULT_BATCH;
        for (Iterator<Tracability> it = tracabilitys.iterator(); it.hasNext();) {
            Tracability tracability = it.next();
            if (tracability.getDate().before(currentDate)) {
                if (tracability.getEvent().equalsIgnoreCase(Constants.BATCH_CONTROL)) {
```

```

        fiability = Constants.FIABILITY_DEFAULT_BATCH;
    } else {
        fiability *= StoreEntities.getWorkstation(
            tracability.getWorkstationID()).
            getFiabilityQualityTask(tracability.getDate());
    }
}
return fiability;
}

public double fiabilityWithQualityTask(MyDate currentDate) {
    double fiability = Constants.FIABILITY_DEFAULT_BATCH;
    for (Iterator<Tracability> it = tracabilitys.iterator(); it.hasNext();) {
        Tracability tracability = it.next();
        if (tracability.getDate().before(currentDate)) {
            fiability *= StoreEntities.getWorkstation(
                tracability.getWorkstationID()).
                getFiabilityQualityTask(tracability.getDate());
        }
    }
    return fiability;
}

public double fiabilityWithQTAndMSLandWorkstation(MyDate currentDate) {
    Tracability previousTracability = null;
    double fiability = Constants.FIABILITY_DEFAULT_BATCH;

    for (Iterator<Tracability> it = tracabilitys.iterator(); it.hasNext();) {
        Tracability tracability = it.next();
        if (tracability.getDate().before(currentDate)) {
            if ((tracability.getEvent().equalsIgnoreCase(Constants.BATCH_CONTROL))
                && (!(previousTracability == null))) {
                fiability = Constants.FIABILITY_DEFAULT_BATCH;
                StoreEntities.getWorkstation(
                    previousTracability.getWorkstationID()).setDateLastControlled(previousTracability.getDate());
            } else {
                fiability *= StoreEntities.getWorkstation(
                    tracability.getWorkstationID()).
                    getFiabilityQualityTaskAndMSL(tracability.getDate());
            }
        } else {
            previousTracability = tracability;
        }
    }
    return fiability;
}

public double fiabilityMSLAndMSESafe(MyDate currentDate) {
    double fiability = Constants.FIABILITY_DEFAULT_BATCH;
    for (Iterator<Tracability> it = tracabilitys.iterator(); it.hasNext();) {
        Tracability tracability = it.next();

```

```

        if (tracability.getDate().before(currentDate)) {
            if (tracability.getEvent().equalsIgnoreCase(
                Constants.BATCH_CONTROL)) {
                fiability = Constants.FIABILITY_DEFAULT_BATCH;
            } else {
                fiability *= StoreEntities.getWorkstation(
                    tracability.getWorkstationID()).
                    getFiabilityMSEAndMSLSafe(tracability.getDate());
            }
        }
    }
    return fiability;
}
}

```

## 15.2. Classe Workstation

```

/*
 * Copyright (C) 2010 Team-WOR (team-war@prunetwork.fr)
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation\th either version 3 of the License\th or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful\th
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not\th see <http://www.gnu.org/licenses/>.
 */
package fr.prunetwork.teamwar.entities;

import fr.prunetwork.teamwar.Constants;
import fr.prunetwork.teamwar.utilities.MyDate;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.Date;
import java.util.Iterator;

/**
 *
 * @author jpierre03+teamwar@prunetwork.fr
 * @author GARCIA Fabien
 * @author NAIT BELKACEM Abdelali
 */
public class Workstation {

    private double fiabilityRandomLaw = Math.random();
    private double disturbanceOfWorkStation =

```

```
        Constants.DISTURBANCE_WORSTATION_PER_EVENT;  
    private String name;  
    private Collection<Tracability> tracabilitys =  
        new ArrayList<Tracability>();  
    private Collection<Tracability> ListOfControls =  
        new ArrayList<Tracability>();  
    private double fiabilityQualityTask =  
        Constants.FIABILITY_DEFAULT_WORKSTATION;  
    double fiabilityTQAndMSL = Constants.FIABILITY_DEFAULT_WORKSTATION;  
  
    public Collection<Tracability> getListOfControls() {  
        return ListOfControls;  
    }  
  
    public void setListOfMSL(Collection<Tracability> setOfMSL) {  
        this.ListOfControls = setOfMSL;  
    }  
    private MyDate dateLastControle = new MyDate();  
  
    public Workstation(String name) {  
        this.name = name;  
    }  
  
    /**  
     * @return the disturbanceOfWorkStation  
     */  
    public double getDisturbanceOfWorkStation() {  
        return disturbanceOfWorkStation;  
    }  
  
    /**  
     * @param disturbanceOfWorkStation the disturbanceOfWorkStation to set  
     */  
    public void setDisturbanceOfWorkStation(double disturbanceOfWorkStation) {  
        this.disturbanceOfWorkStation = disturbanceOfWorkStation;  
    }  
  
    public void addTracability(Tracability tracability) {  
        tracabilitys.add(tracability);  
    }  
  
    /**  
     * @return the tracabilitys  
     */  
    public Collection<Tracability> getTracabilitys() {  
        return Collections.unmodifiableCollection(tracabilitys);  
    }  
  
    /**  
     * @return the name  
     */  
    public String getName() {
```

```

        return name;
    }

    public String description() {
        StringBuilder sb = new StringBuilder();
        sb.append(getName());
        sb.append("\t");
        sb.append("tracabilitys count: ");
        sb.append(getTracabilitys().size());
        return sb.toString();
    }

    public double getFiabilityRandomLaw() {
        while (Double.compare(fiabilityRandomLaw, 0.0) == 0) {
            fiabilityRandomLaw = Math.random();
        }
        return fiabilityRandomLaw;
    }

    /**
     * Fonction which calculate the fiability thanks to the number of events
     * since the last quality task
     * The operation done is this one :
     *   fiabilityQualityTask = Math.max(fiabilityQualityTask,
     *   Constants.FIABILITY_MIN_WORSKATION);
     * @param currentDate
     * @return
     */
    public double getFiabilityQualityTask(MyDate currentDate) {

        fiabilityQualityTask = Constants.FIABILITY_DEFAULT_WORKSTATION
            - getNumberOfEventSinceLastQT(currentDate)
            * Constants.DISTURBANCE_WORSTATION_PER_EVENT;
        fiabilityQualityTask = fiabilityQualityTask
            < Constants.FIABILITY_MIN_WORSKATION
            ? Constants.FIABILITY_MIN_WORSKATION
            : fiabilityQualityTask;
        return fiabilityQualityTask;
    }

    /**
     *
     * @param currentDate
     * @return
     */
    public double getFiabilityQualityTaskAndMSL(MyDate currentDate) {

        fiabilityTQAndMSL = fiabilityTQAndMSL
            - getNumberOfEventSinceLastControl(currentDate)
            * Constants.DISTURBANCE_WORSTATION_PER_EVENT;
        fiabilityTQAndMSL = fiabilityTQAndMSL
    }
}

```

```

        < Constants.FIABILITY_MIN_WORSKATION
        ? Constants.FIABILITY_MIN_WORSKATION
        : fiabilityTQAndMSL;

    return fiabilityTQAndMSL;
}

private Double getNumberOfEventSinceLastQT(MyDate currentDate) {

    Date lastQualityTask;
    Double numberofEvents = new Double(0);

    for (Iterator<Tracability> it = getTracabilities().iterator();
         it.hasNext();)
    {
        Tracability tracability = it.next();
        if (tracability.getDate().before(currentDate)) {
            if (tracability.getEvent().equalsIgnoreCase(
                Constants.QUALITYTASK))
            {
                lastQualityTask = tracability.getDate();
                numberofEvents = new Double(0);
            } else {
                numberofEvents++;
            }
        }
    }
    return numberofEvents;
}

public Double getFiabilityMSEAndMSLSafe(MyDate currentDate) {
    getDateLastControle(currentDate);
    Double nbEvents = getNumberOfEventSinceLastControl(currentDate);
    fiabilityTQAndMSL = Math.max(Constants.FIABILITY_MIN_WORSKATION,
        Constants.FIABILITY_DEFAULT_WORKSTATION - nbEvents
        * Constants.DISTURBANCE_WORSTATION_PER_EVENT);
    return fiabilityTQAndMSL;
}

private void getDateLastControle(MyDate currentDate) {

    MyDate lastQualityTask = null;
    Double numberofEvents = new Double(0);
    Iterator<Tracability> it = getListOfControls().iterator();

    while ((it.hasNext()) && (lastQualityTask == null)) {
        Tracability tracabilityBefor = it.next();
        if (it.hasNext()) {
            Tracability tracabilityAfter = it.next();
            if (tracabilityBefor.getDate().before(currentDate)
                && tracabilityAfter.getDate().after(currentDate)) {
                lastQualityTask = tracabilityBefor.getDate();
            }
        }
    }
}

```

```

        } else {
            if ((tracabilityBefor.getDate().before(currentDate)) {
                lastQualityTask = tracabilityBefor.getDate();
            }
        }

    }
    if (lastQualityTask != null) {
        dateLastControle = lastQualityTask;
    }
}

/**
 * @param currentDate
 * @return
 */
private Double getNumberOfEventSinceLastControl(MyDate currentDate) {
    Double numberofEvents = new Double(0);
    for (Iterator<Tracability> it = getTracabilities().iterator();
         it.hasNext();) {
        Tracability tracability = it.next();
        if ((tracability.getDate().before(currentDate))
            && tracability.getDate().after(dateLastControle)) {
            numberofEvents++;
        }
    }

    return numberofEvents;
}

/**
 * @param lastControleMSE the lastControleMSE to set
 */
public void setDateLastControle(MyDate lastControleMSE) {
    this.dateLastControle = lastControleMSE;
}

/**
 * @return the lastControleMSE
 */
public MyDate getDateLastControle() {
    return dateLastControle;
}
}

```

### 15.3. Classe Marcheur

```

/*
 * Copyright (C) 2010 Team-W@R (team-war@prunetwork.fr)
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or

```

```
* (at your option) any later version.  
*  
* This program is distributed in the hope that it will be useful,  
* but WITHOUT ANY WARRANTY; without even the implied warranty of  
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
* GNU General Public License for more details.  
*  
* You should have received a copy of the GNU General Public License  
* along with this program. If not, see <http://www.gnu.org/licenses/>.  
*/  
package fr.prunetwork.teamwar;  
  
import fr.prunetwork.teamwar.entities.Batch;  
import fr.prunetwork.teamwar.entities.StoreEntities;  
import fr.prunetwork.teamwar.entities.Tracability;  
import fr.prunetwork.teamwar.entities.Workstation;  
import fr.prunetwork.teamwar.extractor.BatchAndWorkstationLinkExtractor;  
import fr.prunetwork.teamwar.extractor.BatchExtractor;  
import fr.prunetwork.teamwar.extractor.WorkstationExtractor;  
import fr.prunetwork.teamwar.storage.reader.ExtractDataFromFile;  
import fr.prunetwork.teamwar.storage.writer.StoreDataToFile;  
import fr.prunetwork.teamwar.utilities.MyDate;  
import java.io.IOException;  
import java.text.DecimalFormat;  
import java.util.Collection;  
import java.util.Iterator;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
/**  
 *  
 * @author jpierre03+teamwar@prunetwork.fr  
 * @author GARCIA Fabien  
 * @author NAIT BELKACEM Abdelali  
 */  
public class Marcheur {  
  
    private String fichier;  
    private StoreDataToFile sdtf;  
    private Collection<Tracability> tracabilitys;  
    private WorkstationExtractor we;  
    private BatchExtractor be;  
    private Collection<Workstation> workstations;  
    private Collection<Batch> batchs;  
  
    public Marcheur() {  
        /fichier = Constants.SHORT_FILE;  
        fichier = Constants.LONG_FILE;  
        sdtf = new StoreDataToFile("./ResultForExcelV2.txt");  
  
        tracabilitys = ExtractDataFromFile.createTracabilityCollection(fichier);  
    }  
}
```

```
we = new WorkstationExtractor(tracabilitys);
workstations = we.extract();

be = new BatchExtractor(tracabilitys);
batchs = be.extract();

BatchAndWorkstationLinkExtractor batchAndWorkstationLinkExtractor =
    new BatchAndWorkstationLinkExtractor();
batchAndWorkstationLinkExtractor.link();
Double maxNumber0fSteps = StoreEntities.getNumberMax0fSteps();
MyDate lastDate = StoreEntities.getLastDate();
MyDate firstDate = StoreEntities.getFirstDate();
System.out.println(firstDate.toString());
System.out.println(lastDate.toString());
int nunmber0fProcessDone = 0;
for (Iterator<Batch> it = batchs.iterator(); it.hasNext();) {
    Batch batch = it.next();

    DecimalFormat decimalFormat = new DecimalFormat();
    decimalFormat.setMaximumFractionDigits(4); //arrondi à 2 chiffres apres la virgule
    decimalFormat.setMinimumFractionDigits(4);
    Double percent0favencement =
        new Double(batch.getTracabilitys().size() / maxNumber0fSteps);
    sdtf.add(decimalFormat.format(percent0favencement).replace(",","."));
    + " " + decimalFormat.format(batch.fiabilityMSLAndMSESafe(lastDate)).replace(",",".");
    System.out.println("Nombre de batch traité :"
        + nunmber0fProcessDone++ + " / " + batchs.size());
}

try {
    sdtf.commit();
} catch (IOException ex) {
    Logger.getLogger(Marcheur.class.getName()).log(Level.SEVERE, null, ex);
}
}

public static void main(String[] args) {
    new Marcheur();
}
}
```



## Bibliographie

- [1] Wikipedia st. [2.1](#)
- [2] Amotz Bar-Noy, Frank K.Hwang, Ilan Kessler, and Shay Kut. A new competitive alogorithm for group testing. *INFOCOM-92*, September 1996. [5.1](#), [12.8](#)
- [3] Belgacem Bettayeb. *Optimized design of control plans based on risk exposure and ressources capabilities*. PhD thesis, S-SCOP laboratory, 2010. [12.3](#)
- [4] Belgacem Bettayeb, Philippe Viallette, Samuel Bassetto, and Michel Tollenaere. *An aproach for operational risk evaluation and its link to control plan*. PhD thesis, G-SCOP laboratory, 2010. [6.1](#), [12.4](#)
- [5] D.Z. Du and F.K Hwang. *Discrete Applied Math*, chapter Competitive Group Testing. AT&T Bell Laboratories Technical Memorandum, December 1990. [5.3](#)
- [6] Mhamed Sahanoun, NouvelAuteur<sup>2</sup>%2, NouvelAuteur<sup>3</sup>%2, Soidri Bastoini, and Michel Tollenaere. *Optimizing Return on inspection trouth defectivity smart sampling*. PhD thesis, INP Grenoble, G-SCOP, 2010. [7](#), [12.5](#)
- [7] Mhamed Sahanoun, Philippe Viallette, Samuel Bassetto, Soidri Bastoini, and Michel Tollenaere. *Computation of wafer-At-Risk from theory to real life demonstration*. PhD thesis, INP Grenoble, G-SCOP, 2010. [10](#), [12.6](#)