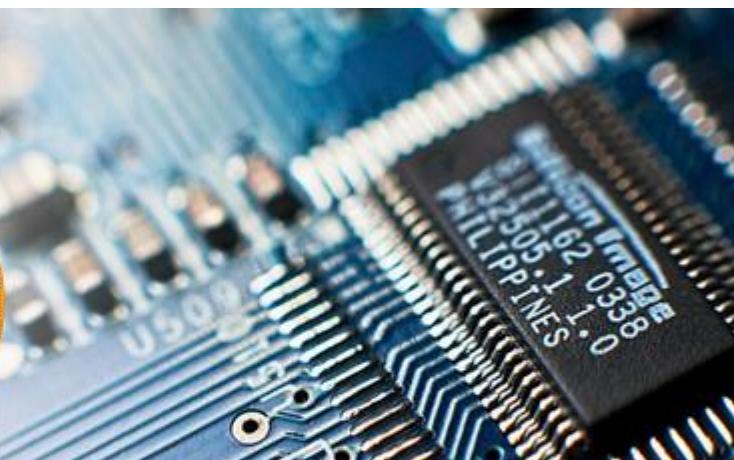


# Rapport de stage 2<sup>eme</sup> année DUT GEII



**BERTHET Vincent**

Développement Java Swing

**Nom de l'entreprise : Mu-TEST**

**Tuteur entreprise : Mr Rochedix**

**Tuteur IUT : Mr Bernard**

**Dates : Du 4/04/2016 au 4/08/2016**

**Année scolaire 2015/2016**

# Notes

---

## Remerciements :

Tout d'abord, je tiens à exprimer ma profonde gratitude à **Mr Ambroise Rochedix**, mon tuteur entreprise, sans qui je n'aurais pas pu réaliser ce stage. Je le remercie notamment pour son accueil, sa confiance, sa pédagogie, sa motivation et pour le temps qu'il m'a accordé, tout cela malgré un planning rigoureux.

Merci de m'avoir initié au Java et de m'avoir considérer comme un membre de l'équipe Software durant la durée de ce stage.

Je tiens également à remercier Monsieur le CEO de la société Mu-TEST, **Mr Mathieu Dupez**, qui m'a laissé l'opportunité de me joindre au sein de son entreprise.

Merci aussi à évidement **tous les membres de Mu-TEST** qui m'ont entouré lors de mon stage. Ceux-ci m'ont apporté de nombreuses connaissances aussi bien professionnelles, qu'humaines. Grâce à eux, j'ai pu découvrir le monde professionnel de ma formation.

Ce stage à leur côté m'as permis de m'enrichir considérablement personnellement et de renforcer ma vision sur ma poursuite d'étude.

De plus, je remercie mon professeur référent IUT **Mr Florent Bernard** pour m'avoir supervisé lors du déroulement de mon stage et avoir pris le temps de me rendre visite chez Mu-TEST où il a été très intéressé par l'entreprise et le contenu de mon stage.

Enfin, pour conclure, je souhaiterais remercier **toutes les personnes qui ont participé** de différentes façons à la réussite de mon stage.

Vincent Berthet

# Sommaire

---

<b>I. Abstract</b>	p4
<b>II. Introduction</b>	p5
<b>III. Mu-TEST</b>	p6
A. Présentation	p6
B. Les composants électroniques	p8
C. L'ATE	p9
D. Présentation du système de Mu-TEST	p10
E. Suite Logicielle	p13
F. Objectifs de Mu-TEST : « The ATE Game changer »	p18
<b>IV. Organisation</b>	p19
G. Organigramme	p19
H. Les départements	p20
I. Méthodologie de travail : SCRUM	p21
J. Environnement de travail	p22
<b>V. Mon activité</b>	p24
K. Maintenance	p24
L. Projet	p31
<b>VI. Conclusion</b>	p50
<b>VII. Annexes</b>	p51
<b>VIII. Glossaire</b>	p62

# I. Abstract

---

J'ai poursuivi mon stage de fin d'étude en génie électrique et informatique industrielle au sein de la société **Mu-TEST** (à Saint-Just-Malmont). Cette société conçoit équipement de test automatique (ATE) qui sont des testeurs. Ceux-ci permettent d'exécuter le contrôle de caractéristiques de composants électroniques. Au cours de mon stage j'ai donc découvert le milieu de l'ATE dans le détail. De plus j'ai poursuivi un projet de développement Software afin de répondre à un besoin de l'entreprise : la mise en place d'un **système de Tag**. Pour cela j'ai dû dans un premiers temps apprendre le langage **Java** ainsi que le fonctionnement de l'architecture Software et les outils de développement de l'entreprise. En parallèle de ce développement, j'ai également réalisé des maintenances préventives des fonctionnalités du testeur et de ses logiciels.

Au cours du mon activité au sein de l'entreprise j'ai dû suivre la méthodologie de travaux en place : SCRUM, ce qui fut tout nouveau pour moi. La réalisation de mon stage au sein de Mu-TEST m'as permis de m'enrichir en tous points et de consolider mon idée de poursuite d'étude de l'informatique en école d'ingénieurs.

I have done my internship of end of study in electrical engineering and industrial computing

This company designs Automated Test Equipment (ATE), in ATE field they are called testers. They are used to run test control of electrical component features.

During my internship, I have discover the ATE field in the detail. Moreover I have managed a Software development project in order to reply at a need of the company: the creation of a **Tag system**. In this order, in a primary time, I had to learn the Software language **Java** and the same way the Software architecture and the developments tools of the company. In parallel of this project, I have also to realize preventive maintenance of the tester and software features.

Also, during my activity within the company, I have to follow the work way of **Mu-TEST** : SCRUM, whick was a new way of work for me. The achievement of my internship inside of this company, able me to extend myself in each points and to improve my view of my study pursuit in a software engineering school.

## II. Introduction

---

Dans le cadre de ma formation au Diplôme Universitaire Technologique en Génie Electrique et Informatique Industriel (GEII), je suis amené à effectuer un stage de douze semaines dans le monde professionnel.

Durant cette période d'activité, je me dois en tant que professionnel de réaliser les tâches qui me sont affectées. De plus, il faut que je poursuive un projet. Ce projet doit répondre à un besoin de l'entreprise et permettre d'exploiter les notions et compétences acquises tout au long de la formation.

Etant à la recherche d'un projet qui correspondrait à mes attentes, à savoir plutôt dans l'informatique et orienté vers une poursuite d'étude en master, j'ai donc contacté de nombreuses entreprises prêtes à m'accueillir.

C'est une entreprise basée en Haute-Loire (à Saint-Just-Malmont) qui m'a offert la chance de poursuivre un projet correspondant à mes attentes : **Mu-TEST « The ATE game changer »**

**Mu-TEST** est une start-up qui se montre innovante dans le domaine du test électronique, qu'on appelle plus communément dans ce milieu l'*ATE<sub>1</sub>*.

Lors de mon stage, (de quatre mois me concernant), j'ai pu donc intégrer une jeune entreprise novatrice, en plein essor dans un marché fermé qu'est le test de composants électroniques, un marché d'avenir.

1. *ATE* : *Automated Test Equipment* : équipement de test automatique, système permettant de soumettre un composant électronique à différents test pour définir si les caractéristiques sont correctes ou non.



Les mots *en vert* sont référencés dans le Glossaire.

### III. MU-TEST

---

#### A. Présentation

**Mu-TEST** est une société récente, fondée seulement en 2009. L'entreprise poursuit depuis un fort développement, ceci est notamment lié au fait qu'elle dispose d'expériences locales non négligeables.

En effet, suite à la désindustrialisation du bassin stéphanois depuis les années 90, de nombreuses entreprises ont été en difficulté économique. Le bassin stéphanois comportait alors quelques rares entreprises européennes sur le marché de l'ATE.

Certaines entreprises ont fermé. C'est notamment le cas du sur le site de Saint-Etienne d'une grande entreprise française dans l'électronique, **Schlumberger**, mettant fin à tout développement dans l'ATE.

**Schlumberger**



C'est en décembre 2009 que **Mathieu Duprez** avec une expérience de plus de 25 ans dans l'ATE décida de fonder une SAS, l'entreprise Mu-TEST.

Implantée en Haute-Loire (Saint-Just-Malmont), l'entreprise est constituée de membres piliers issus de Schlumberger, ce qui est l'une des raisons de sa réussite.

A ce jour, l'entreprise comporte dix-huit employés dont trois alternants !



Figure 1 : Locaux de MuTest

Mu-TEST est une entreprise qui conçoit et produit des structures complètes pour le test de composants électroniques, que ce soit en petit ou gros volumes. L'entreprise traite avec des clients se situant aux quatre coins du globe.

Le test de composants électroniques est essentiel pour les fabricants électroniques afin de déceler les imperfections de la production. Les perspectives d'avenir du marché de l'ATE sont liées à celui du développement de l'électronique, ce qui présente donc un fort potentiel.

Suite au fort développement de l'entreprise, son chiffre d'affaire a beaucoup augmenté. Il était de 600 000 € en 2013, et de 1,2M € en 2014, preuve de son développement important.

## B. Les composants électroniques

De nos jours, on ne peut passer à côté des composants électroniques. Le monde autour de nous est désormais constitué d'électronique.

Ceci est dû à la forte avancée de l'industrie électronique qui a permis d'élaborer à grande échelle des composants électroniques de plus en plus complexes et précis.

Les composants électroniques sont conçus en groupes sur des plaques de silicium (appelées Wafer [gaufre en anglais]).

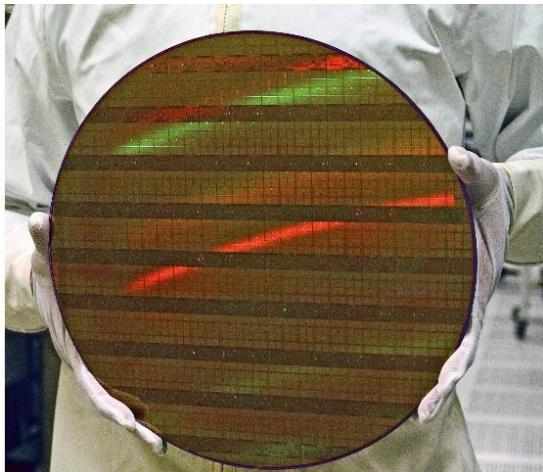


Figure 2 : Un Wafer

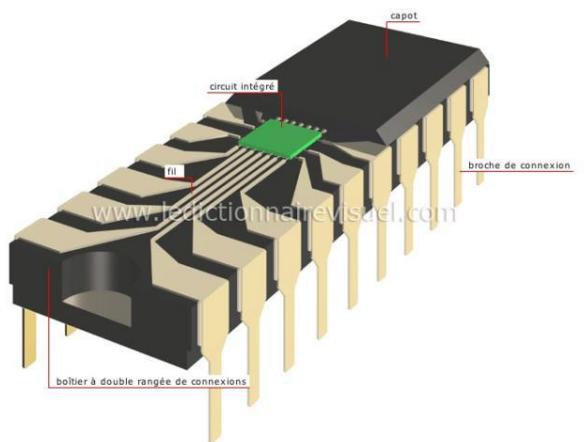


Figure 3 : Constitution d'un composant

Les wafers sont ensuite découpés au laser afin d'obtenir des puces qui seront mises à l'intérieur d'un boîtier tel que nous le connaissons.

Cependant, toute production n'étant pas parfaite, il est essentiel de contrôler la conformité du produit le plus rapidement et efficacement possible en *salle blanche*<sub>2</sub> : c'est là qu'intervient l'**ATE**.

2. *Salle Blanche* : Pièce où la concentration particulière est maîtrisée afin de minimiser l'introduction, la génération, la rétention de particules au sein de la pièce

## C.L'ATE

L'ATE permet de tester les circuits à leur plus bas niveau de conception, selon un plan de test défini qui soumet et récupère différentes tensions et intensités afin de répondre à ses caractéristiques.

Ce procédé représente la solution la plus rapide et efficace afin de tester des composants à grande et petit échelle.

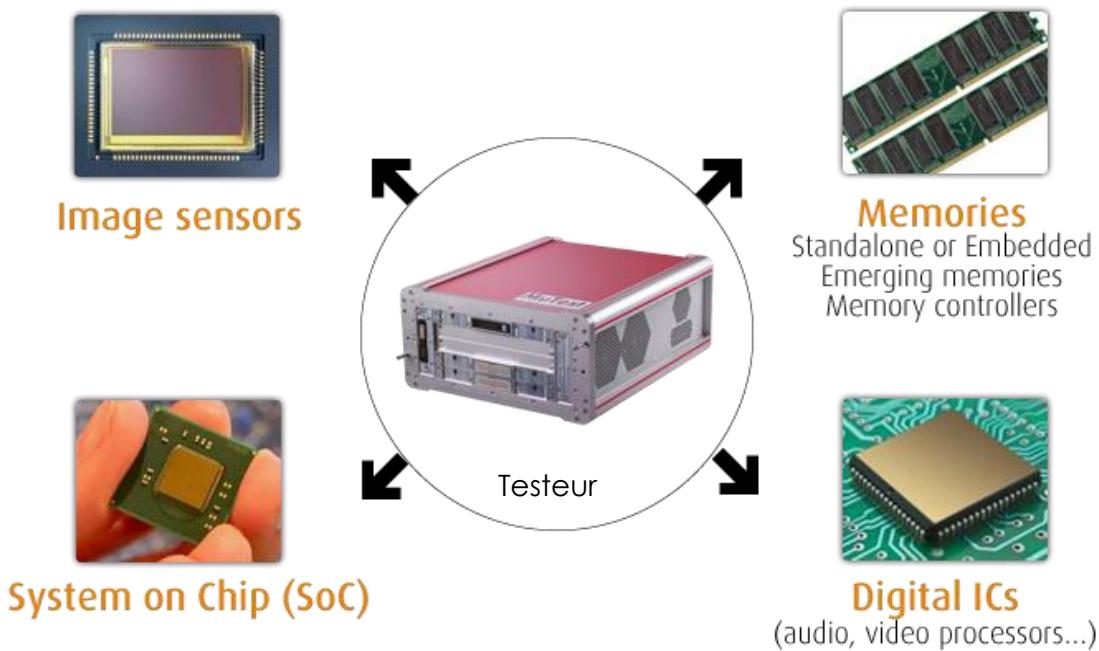


Figure 4 : Rayon de test

Les *DUT*<sub>3</sub> sont physiquement connectés au testeur par une autre machine (*handler*<sub>4</sub> / *prober*<sub>6</sub>), cela permet de manipuler les composants rapidement.

Les résultats du test sont affichés de façon visuelle, on appelle cela une wafermap.

Il faut savoir que pour les industriels, le test d'un composant représente en moyenne 10 % du prix de revient, ce qui n'est donc pas négligeable.

3. *DUT* : Device Under Test : dispositif à tester, ici les composants électroniques produits

4. *Handler* : maître : bras robot qui place automatiquement les composants dans un *socket*<sub>5</sub>.

5. *Socket* : cavité : conteneur où l'on dispose un composant, les sockets varient en fonction des composants

6. *Prober* : sonde : utilisation de sondes qui vont se connecter directement sur le wafer.

## D. Présentation du système de Mu-TEST

- Le produit :



Figure 5 : Système « M5S » de Mu-TEST

Il est commandé par une carte « *Gateway*<sub>8</sub> » qui s'avère être un serveur Linux. Cette carte constitue l'interface entre l'ordinateur et le système. La communication s'établit via une simple liaison Ethernet ou RS232.

Le **M5S**, est le système de « base »,

Il doit son nom au fait qu'il puisse loger jusqu'à cinq instruments. Cela permet en conséquence d'en faire un système très modulable. Il existe aussi des versions **10S** et **21S** ([Annexe Système A1 A2](#))!

Ces grands atouts sont :

- ✓ Sa modularité
- ✓ Son encombrement ([Système A3](#))
- ✓ Sa ventilation (Là où les concurrents utilisent un [water cooling](#))



Figure 6 : 2 LoadBoard (M10S - Multicartes en haut)

Sur la face avant du système se fixe mécaniquement le « *LoadBoard*<sub>9</sub> ».».

Il existe différents types de LoadBoard en fonction des caractéristiques du composant à tester (socket...). Un LoadBoard multicartes a également été conçu en interne afin d'effectuer plus facilement des tests.

Celui-ci permet de faire la liaison entre le composant qui sera testé et le testeur (via les [pogo pins](#)<sub>10</sub> des instruments).

7. *Water cooling* : refroidissement du système via un système de circulation d'eau froide

8. *Gateway* : Portail : Désigne la passerelle réseau permettant de relier deux réseaux distincts.

9. *LoadBoard* : Plateau de chargement : outils qui permet de réaliser l'interface entre le composant et le testeur.

10. *Pogo pin* : fine pointe en or qui réalise un contact précis et efficace entre le LoadBoard et le testeur

## • Les instruments :

Les instruments constituent le fonctionnement du système. En effet, ce qui différencie chaque système sont les instruments utilisés.

Certains sont utilisés pour des signaux numériques uniquement (**M-D864 / M-D1632**), signaux mix (**M-MiXW**), pour des grandes tensions (**M-WR48** [marché automobile]) ou bien encore tout simplement pour alimenter (**M-DPS10**) le système.

Ce qui fait la force de ces instruments, c'est que contrairement à la concurrence ils sont composés de composants « communs » à trouver. Mais la grande particularité de Mu-TEST face à ses concurrents et que son système repose sur des *FPGAs*<sup>11</sup>. Là, où ceux de la concurrence sont composés d'un processeur dédié (étudié spécialement pour).

Mu-TEST propose les derniers FPGAs du marché. Cela permet :

- ✓ une évolution du système dans le temps (contrairement à un processeur)
- ✓ de bonnes performances
- ✓ de baisser fortement les coûts (non développement d'un processeur spécifique)
- ✓ une faible consommation (et donc un refroidissement plus faible du système)

### Test instruments

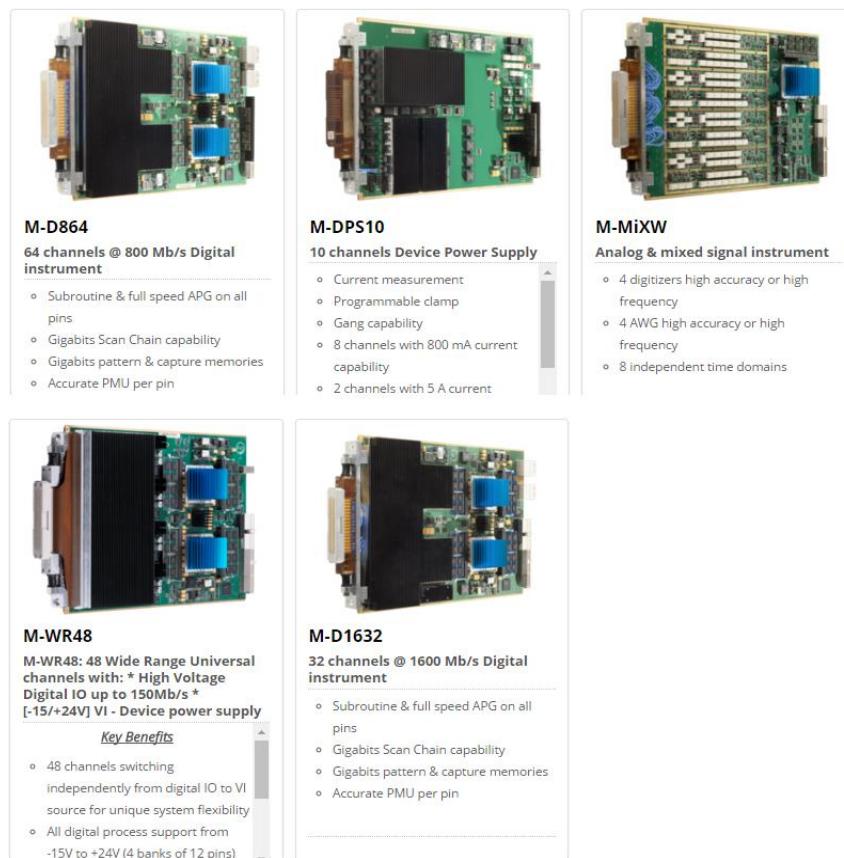


Figure 7 : Les différents instruments – [www.Mu-TEST.com/catalogue/mt19s](http://www.Mu-TEST.com/catalogue/mt19s)

11. *FPGA* : Field Programmable Gate Array : circuit Logique Programmable, cela permet d'appliquer des niveaux en sortie en fonctions des entrées dans le but de commandé les instruments du testeur.

Chaque Testeur étant relié au réseau Ethernet via la **Gateway**, il dispose en conséquence d'une adresse IP (statique). C'est avec cette IP que l'on contrôlera notre Testeur.

L'accès à cette adresse IP nous permet de visualiser diverses informations sur le Testeur :

- ✓ les instruments utilisés
- ✓ les slots des instruments
- ✓ l'état des instruments
- ✓ les versions de codes FPGAs de chaque instrument
- ✓ les utilisateurs actifs sur le système
- ✓ ....

Slot	Type	UID	Status	Resource	BUS/ID	Code
00	M-GTW		ON OK		1/65	15101510
01	M-D864	21A2-C0DE-C0DE-C0DE	ON OK	CTL0: 0x0100 CTL1: 0x0120	CTL0: 1/66 CTL1: 1/67	CTL0: 16052302 CTL1: 16052302 CTL2: 15102201
02	M-MIXW	3D30-C0DE-C0DE-C0DE	ON OK	CTL0: 0x6000	CTL0: 1/68	CTL0: 16030901
03	M-D864	01B5-C0DE-C0DE-C0DE	ON OK	CTL0: 0x0140 CTL1: 0x0160	CTL0: 1/69 CTL1: 1/70	CTL0: 16052302 CTL1: 16052302 CTL2: 15102201
05	M-DPS10	031B-C0DE-C0DE-C0DE	ON OK	CTL0: 0x4000	CTL0: 1/71	CTL0: 16030701

```

U-Boot 1.0 MuTest (2012-06-06 - 15:37:04 CEST) MT800
System: Linux 2.6.32.3#6 2012-07-26 12:22:50 CEST
mt800gtw has been running for: 18:35:23 up 3 days, 18:35, load average: 0.00, 0.00, 0.00
MT800GTW: Server Compiled on 02/26/16 11:56:15

```

## Connections

Tool	User	Host	ID
Viewer	vberthet	TOSHIBA01MOB	35

Figure 8 : L'état du système ayant l'IP 192.168.0.228

## E. Suite Logicielle

Afin de pouvoir commander les testeurs, Mu-TEST a développé toute une **plateforme logicielle** intuitive installable sur les systèmes d'exploitation Windows.

Ces outils permettent de visualiser et de commander le testeur à distance via une liaison Ethernet ou RS232.

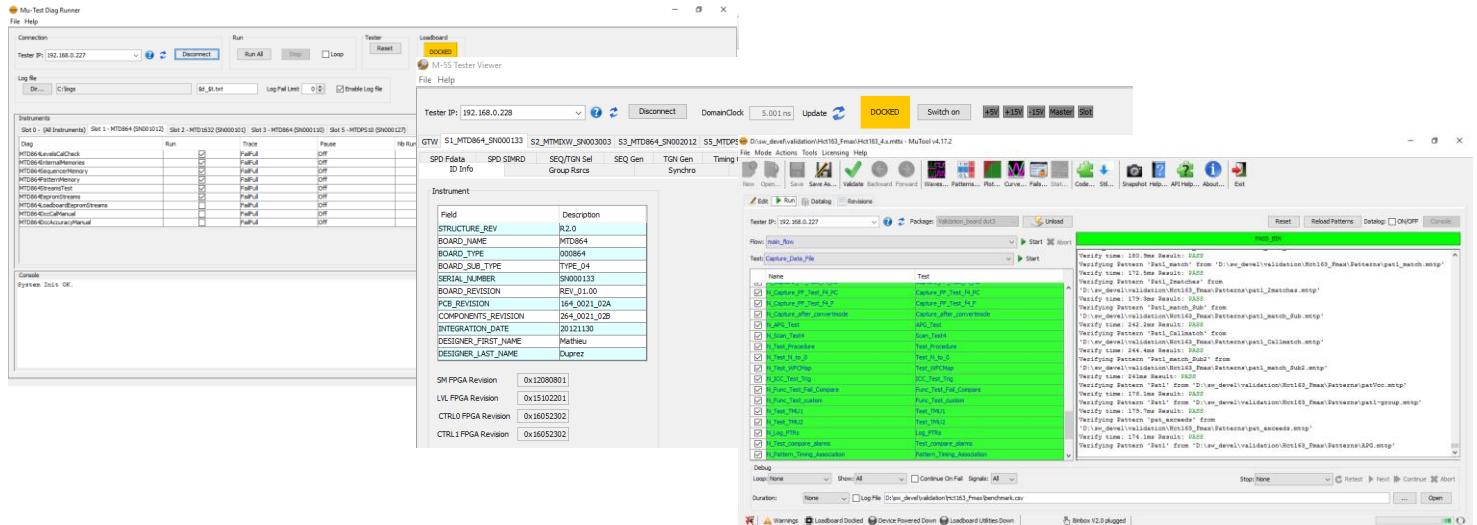


Figure 9 : Un aperçu de la suite logicielle

Toute la suite des interfaces graphiques est développée en Java.  
Cependant, on retrouve également de nombreuses couches de  
différents langages, tels le :

- **C**, utilisé pour la communication entre l'interface graphique et le testeur
- **STILL (Standard Test Interface Language)**, qui définit les *patterns*<sup>12</sup> des tests.
- **XML**, pour retranscrire le contenu d'un programme de test.
- **DLL**, qui constitue l'user code
- **Python**, utilisé pour la partie FPGAs

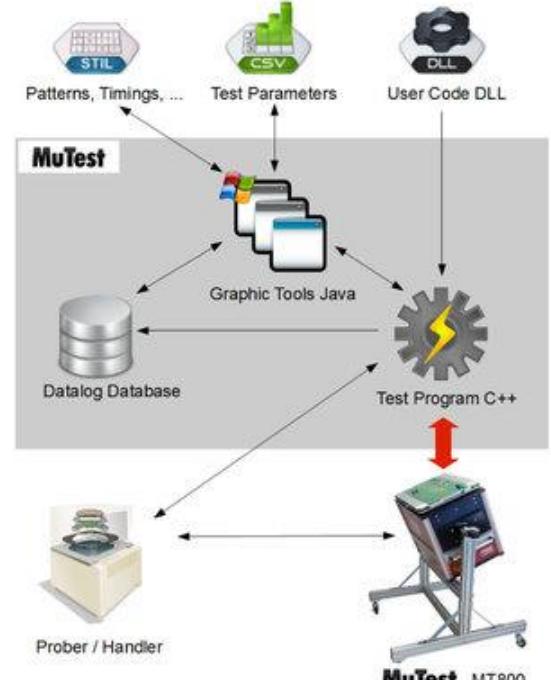


Figure 10 : Schéma fonctionnel d'un Testeur

12. *Pattern* : modèle : solution générique à un comportement. C'est en réalité une table des vérités qui permet d'appliquer un résultat en sortie en fonction des entrées.

## • MuTool : Gestionnaire de test

**MuTool** permet de configurer toute la gestion des tests. C'est avec cette interface que nous allons rédiger/sauvegarder les tests que nous voulons réaliser.

Il s'agit de l'interface le plus utilisé par le client et donc le plus complet en termes de fonctionnalités. On y retrouve de nombreux outils afin d'exploiter l'exécution d'un test.

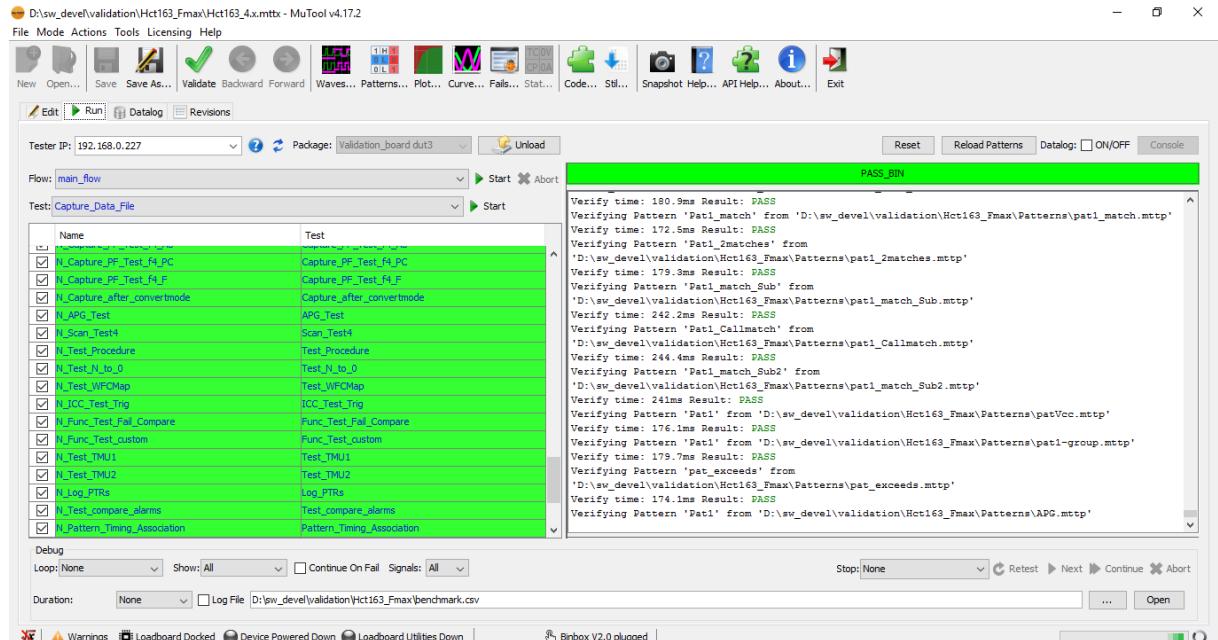


Figure 11 : Un test d'un compteur qui est conforme

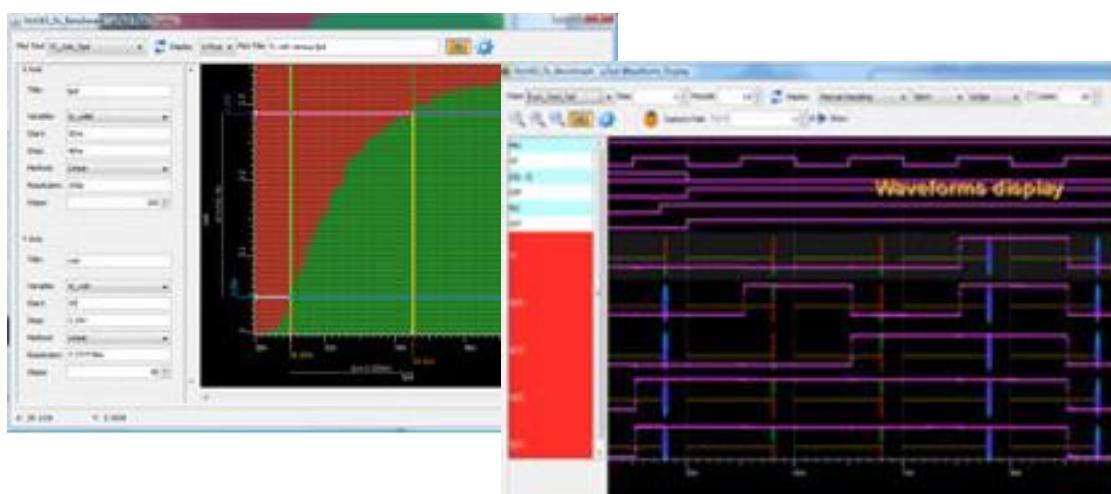


Figure 12 : Le WaveForm display permet de visionner la forme des signaux

## • MuDiagnostic : Calibration des testeurs

Les besoins des clients requérant une certaine qualité et exactitude des mesures, il est important que chaque testeur soit correctement calibré en fonction de ses caractéristiques et des FPGAs de la Release.

Ainsi, chaque instrument inséré dans un slot du système doit pouvoir être calibré indépendamment afin de garantir l'exactitude des sorties, que ce soit sur l'échelle temporelle ou de tension/intensité.

La qualité et la fonctionnalité de ce système est critique : c'est une garantie d'exactitude et de qualité pour les clients de Mu-TEST. C'est l'élément qui assure l'exactitude des résultats de test du client, sur ses composants.

Cet outil permet de réduire grandement l'incertitude en sortie des instruments afin de rentrer dans des spécifications très précises à soumettre aux composants.

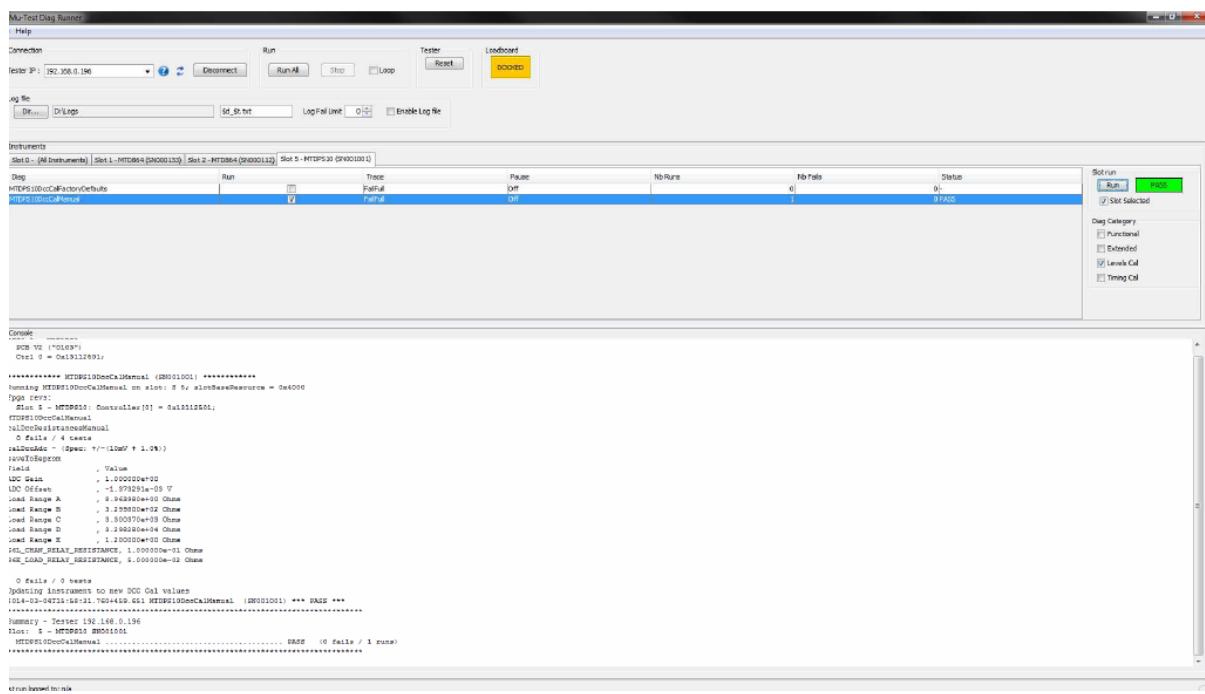


Figure 13 : Système de Diagnostique

- Viewer : Visualisation des registres

Ce système est conçu pour les étapes d'engineering et de vérification. Il permet de visualiser, modifier des registres du testeur. Il existe tout une panoplie de **Viewer** afin d'observer différentes valeurs, l'intérêt de ceux-ci est qu'ils ne bloquent pas l'accès au testeur.

En effet, contrairement à **MuTool** et **MuDiagnostic** et à d'autres outils, on peut charger plusieurs **Viewers** sur un même système.

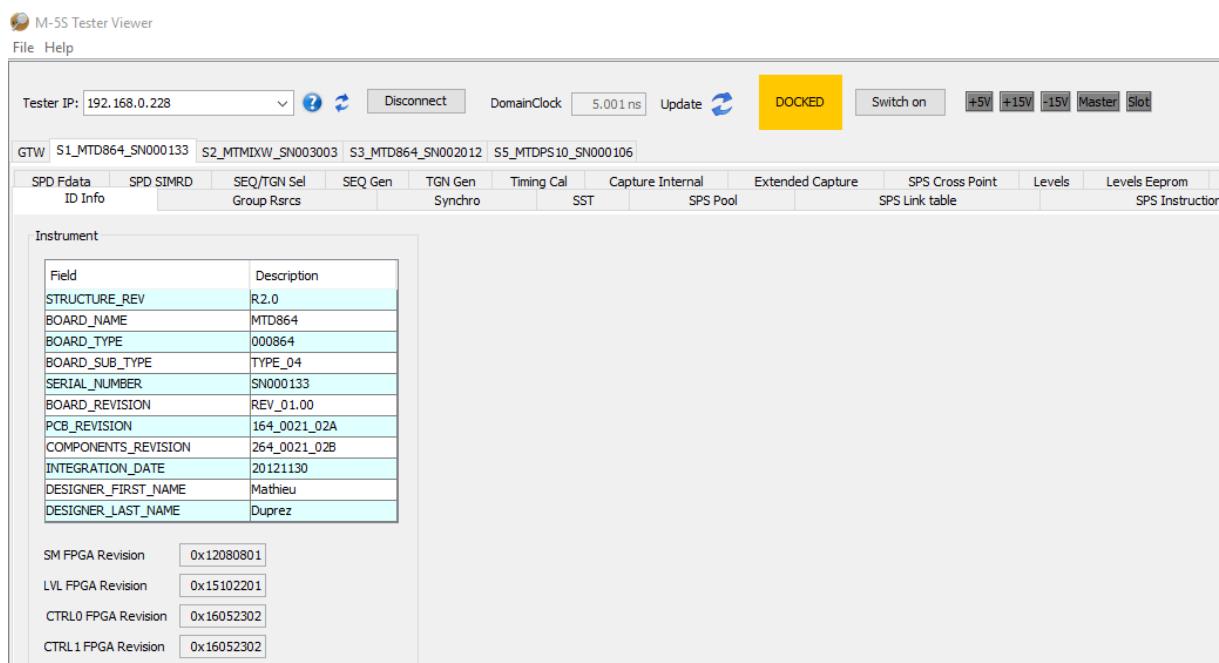


Figure 14 : Exemple d'un Viewer

C'est un outil très pratique, qui permet notamment de contrôler :

- ✓ l'état des instruments
- ✓ la calibration actuelle
- ✓ les registres des instruments
- ✓ les signaux
- ✓ l'alimentation

- Mulnit : Chargement des codes FPGAs

Les FPGAs étant le cœur du système, il existe donc un outil spécifique. Cet outil permet de charger les codes FPGAs dans la *Release*<sup>13</sup> exécuté, ou bien d'en charger d'autres spécifiques.

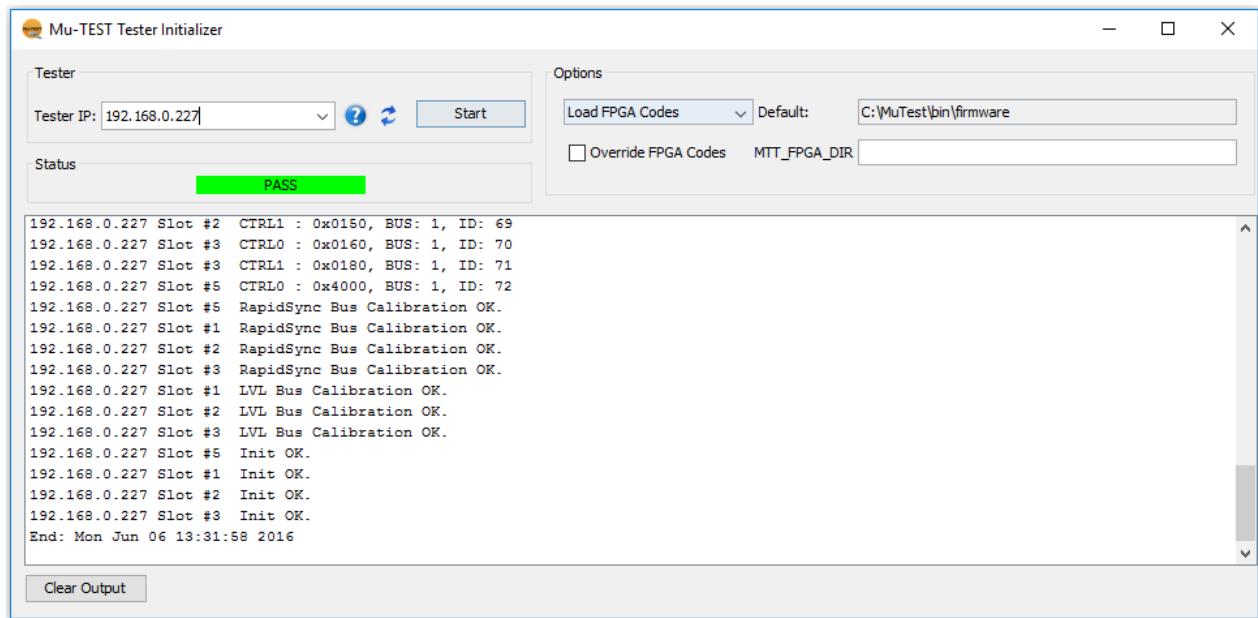


Figure 15 : Chargeur de FPGAs

Outre ces logiciels, il en existe aussi d'autres, tels le :

- **MuProdLauncher** : Il permet de sauvegarder la calibration d'un testeur dans une release précise. Cela évite une nouvelle exécution de la calibration si on souhaite changer de release (changement de FPGAs).
- **MuReleaseSwitcher** : Il modifie la Release utilisée par défaut.
- **Divers scripts** : Ils offrent la possibilité d'effectuer des contrôles sur l'état du système. Certains sont utilisés dans plusieurs programmes de la suite.

13. *Release* : désigne la version d'un logiciel

## F. Objectifs de Mu-TEST : « The ATE Game changer »

En Europe, les constructeurs de testeurs sont très rares. Les compagnies développant des testeurs sont essentiellement des multinationales ayant exporté une partie de leurs productions ou de leurs bureaux de recherches. Le **marché de l'ATE est dominé** à 50% par des entreprises américaines ainsi que 50 % par des entreprises japonaises.

Devenir un acteur important de l'industrie de l'ATE mondial est donc un beau défi pour une jeune firme comme Mu-TEST.

La vision de Mu-Test sur ce que doit être l'ATE afin de s'imposer sur le marché est la suivante :

The screenshot shows the Mu-TEST website's "Vision & Missions" section. At the top, there is a navigation bar with links for Home, Welcome / who we are (which is highlighted in orange), Products, Market & applications, News and events, and Contact. Below the navigation, a breadcrumb trail shows "Home / Vision & Missions". The main heading is "Vision & Missions". To the left of the text, there is a photograph of a rowing team in a boat. The text under "Vision:" states: "Introduce a game changer and unrivalled performance /cost trade-off in the ATE (Automated Test Equipment) world". The text under "Missions" lists several goals:

- Reduce high end tester equipment capex by more than 2X
- Cut OPEX (power, space...) to lower Cost of Test by 30 to 50%
- Bring an innovative high end ATE market offering
  - Desktop solution
  - Configurable & customizable firmware (FPGAs)
  - Plug and play, easy to use systems (universal channels, « monkey proof » Software)

Figure 16 : [www.Mu-TEST.com](http://www.Mu-TEST.com) – La vision de Mu-TEST sur l'ATE

De par son innovation au niveau des systèmes de tests (performance, encombrement, énergie, prix, évolution), Mu-TEST a aujourd’hui acquis en peu d’année la confiance de fabricants électroniques, de grands instituts reconnus aux quatre coins du globe :



**THALES**

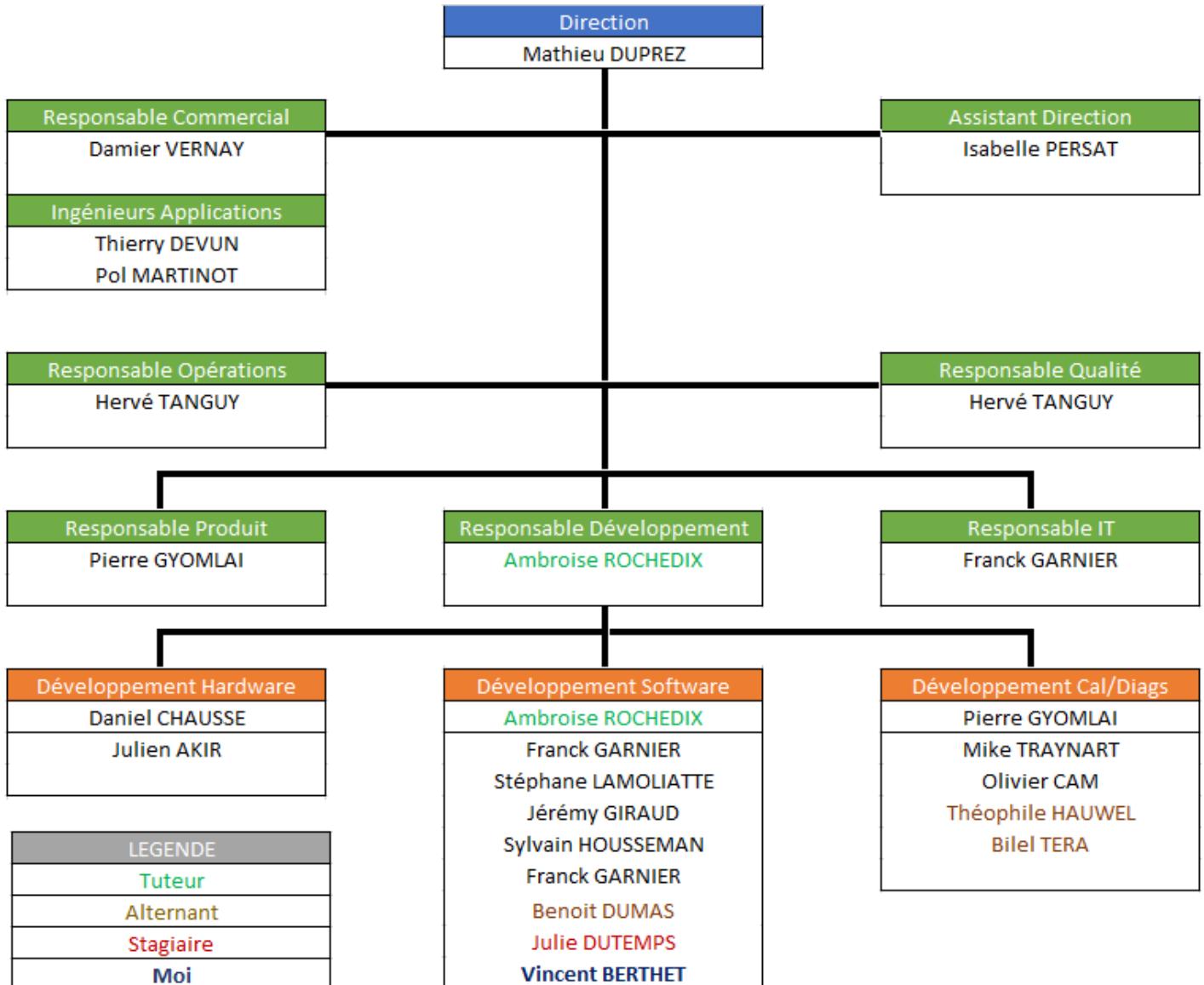


**CISCO**™



# IV. Organisation

## G.Organigramme



J'ai donc intégré le service Développement Software / Maintenance.

## H. Les départements

A l'exception de la partie commerciale et qualité, la totalité du bureau d'études est consacrée au R&D.

On peut découper l'activité du bureau comme suit :

- Département Software (où j'exerce) :

Ce département est composé environ de la moitié de l'effectif des employés de l'entreprise. Ces personnes sont chargées du développement *Software<sub>14</sub>*. Leur but est de développer les logiciels de l'entreprise permettant de réaliser l'interface entre l'opérateur et le testeur afin d'exploiter le système.

- Département Hardware / Firmware:

Le département *Hardware<sub>15</sub>* est composé de deux personnes chargées du fonctionnement du testeur et de la programmation bas niveau (FPGA), le *Firmware<sub>16</sub>*. Elles sont la base du système, ce sont elles qui permettent de soumettre les composants aux tests souhaités. Il s'agit de l'interface entre le Software et le composant.

- Département Ingénieur application :

Il est formé de deux personnes. Celles-ci sont responsables du support client. Elles représentent la vitrine de l'entreprise. En effet, ce sont elles qui vont former, communiquer, réaliser la maintenance vis-à-vis du client. Etant les personnes les plus proches des clients, elles soumettent des évolutions du système afin de répondre aux besoins des clients.

- Département Vente :

Ce département est composé de deux personnes ayant pour objectifs de veiller à la relation avec les sous-traitants de l'entreprise ainsi que de promouvoir la marque à des clients potentiels

14. *Software* : partie logiciel, programmation avec des langages de développement (C, Java ...)

15. *Hardware* : partie matériel, composante du testeur

16. *Firmware* : partie programmation bas niveau via les FPGAs

## I. Méthodologie de travail : SCRUM

Le processus de développement de Mu-TEST suit la méthode « **SCRUM** ». Cette méthodologie est appliquée dans tout le bureau de R&D, à savoir les départements Software, Hardware, « Calibration & Diagnostique ».

Il s'agit d'une méthodologie appliquée à l'informatique, basée sur la méthode *Agile*<sup>17</sup>

Pour simplifier, Scrum se base sur les piliers suivants :

- ✓ des **exigences définies** à atteindre au début du processus (BackLog)
- ✓ des **équipes autonomes et responsables** qui se sentent mobilisées
- ✓ une forte **communication** entre les différentes équipes lors du processus (*mélée quotidienne*<sup>18</sup>)
- ✓ de la **souplesse**, des exigences peuvent être redéfinies au cours du processus
- ✓ un processus de **développement sur trois à quatre semaines** (Sprint)
- ✓ une **réponse aux besoins immédiats** en fin de sprint ; les objectifs accomplis et à réaliser sont définis au cours d'une réunion rassemblant tout le R&D ainsi que le service des ventes et le support.
- ✓ un **produit final fonctionnel**

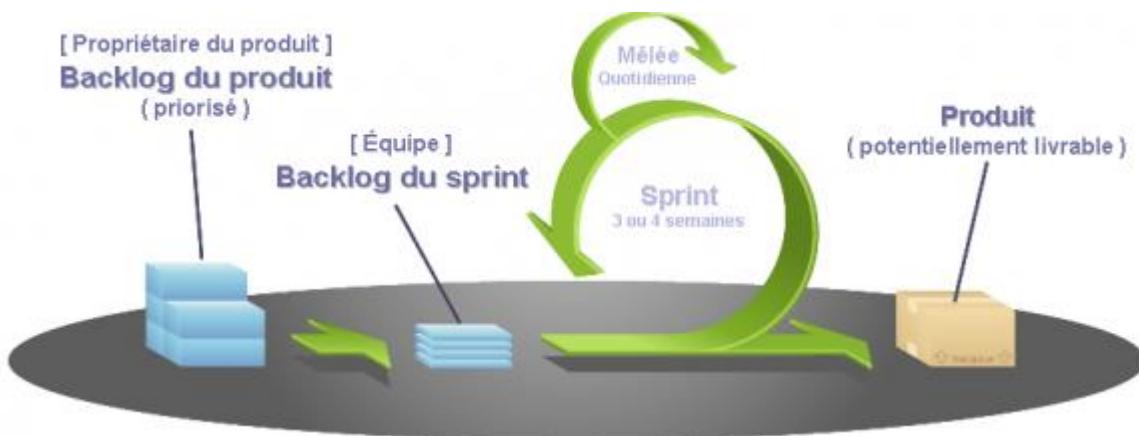


Figure 17 : Représentation graphique de la méthodologie Scrum

Cette méthode privilégie la qualité du produit et le suivi des tâches effectuées. En effet, elle permet de répondre le plus rapidement possible aux priorités de développement de l'entreprise en fonction des besoins du client. C'est une méthode réactive.

Le point fort de cette méthode est de maintenir toutes les équipes informées de l'avancée du développement. La communication est une part très importante de cette méthode.

17. *Méthode Agile* : une méthode agile est une approche itérative et incrémentale ; les tâches vont s'effectuer petit à petit, par ordre de priorité, avec des phases de contrôle et d'échange avec le client.

18. *Mélée quotidienne* : court rassemblement équipe par équipe hebdomadaire afin de réguler les tâches du sprint en cours

## J. Environnement de travail



Figure 18 : Les tâches à réaliser lors du Sprint

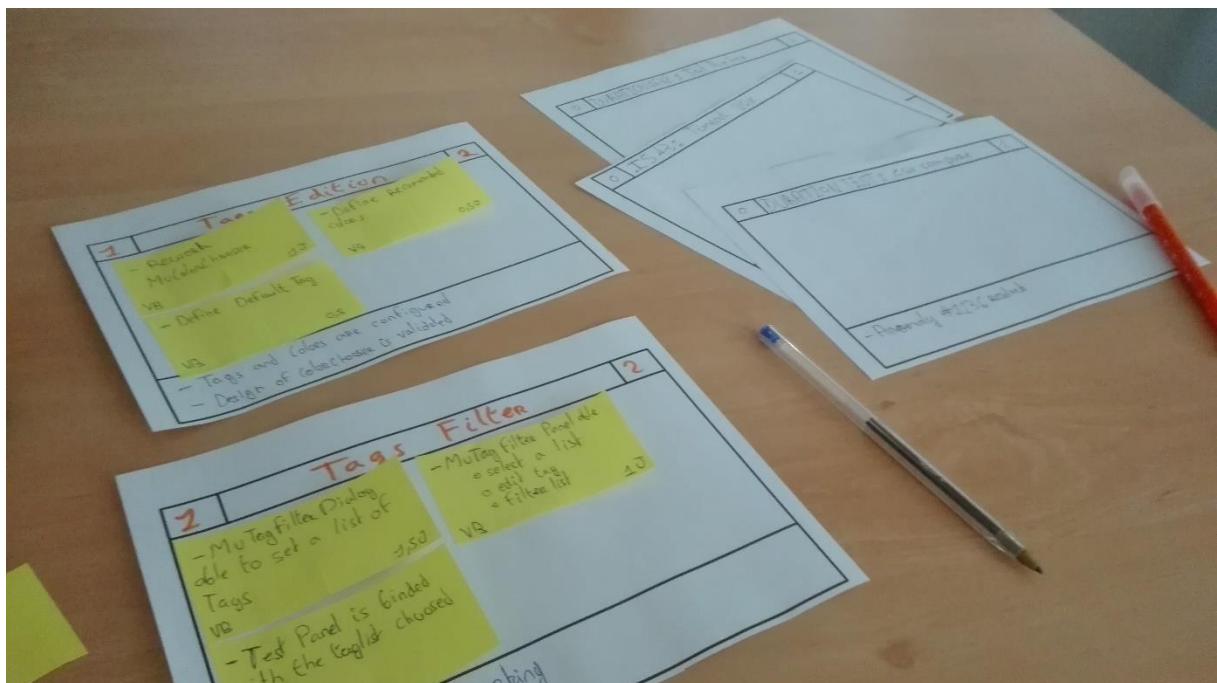


Figure 19 : UserStory – exemple des tâches

L'entreprise étant composé en majeur partie de cadre, les horaires de travail sont plutôt libres.

Les heures recommandés sont de 9h à 12h et 13h30 à 17h30.

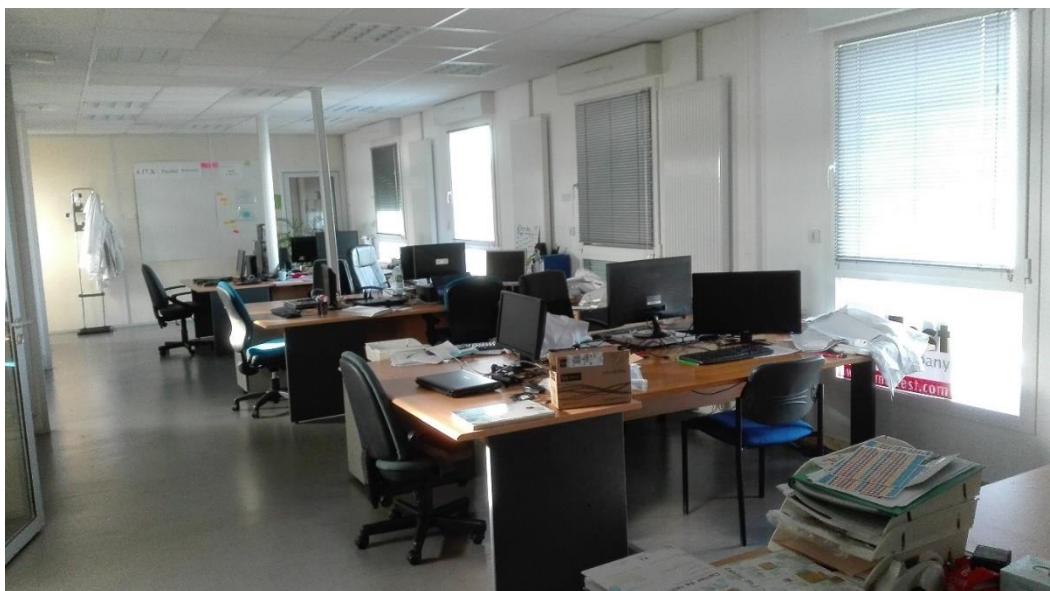


Figure 20 : Openspace, département Software



Figure 21 : Aperçu d'un laboratoire

# V. Mon activité

---

## K. Maintenance

- Importance de la tâche

Comme le veut la méthodologie Scrum, à chaque fin de Sprint le produit livrable doit être fonctionnel.

Dans notre cas, cela signifie que le système est stable à tous les niveaux, c'est-à-dire :

- ✓ le testeur est stable
- ✓ les logiciels sont stables
- ✓ le système est rétro-compatible (les programmes d'ancienne Release sont compatibles)
- ✓ la calibration des instruments corrects



Au cours de son développement il y a de fortes chances que le produit devienne instable ou bien que l'on constate des *Régressions*<sup>19</sup>.

Or, pour l'image de l'entreprise vis-à-vis des clients, il n'est pas concevable de proposer à ses clients une *Update*<sup>20</sup> de sa suite logicielle inférieure voire inutilisable par rapport à la version antérieure utilisée par ses clients. Une analyse préventive est donc pratiquée : la maintenance.

La maintenance représente une tâche importante à ne pas négliger car l'entreprise joue son image et se doit de maintenir la confiance en ses systèmes pour ses clients.

19. *Régression*: désigne des fonctionnalités qui étaient antérieurement fonctionnelles ne le sont désormais plus.

20. *Update*: mise à jour du système, ajouts de nouvelles fonctionnalités, correctifs pour augmenter la stabilité

- Le Processus de validation

Afin d'éviter une majeure partie de potentiels problèmes que peuvent rencontrer les clients, un **processus de validation de Release** est utilisé au sein de l'entreprise.

Durant le déroulement de mon stage, j'ai donc été en charge du processus de validation, un processus important.

### 1. Introduction

Au cours du processus de développement, la suite logicielle subit de nombreuses modifications, permettant d'apporter des fonctionnalités, d'en modifier des existantes, d'en préparer pour de futurs projets, ou bien tout simplement d'améliorer la stabilité du système avec des Fixes (correctifs).

Je suis arrivé au début mois d'avril, dans une période coïncidant avec la fin d'un Sprint qui avait donc débuté il y a trois à quatre semaines. Comme à chaque fin de sprint, un **produit fonctionnel livrable doit être libéré aux clients**. On m'a donc présenté la Release actuel :



MuTest-4-17-1-RC9-  
10-04-2016.exe

Figure 22 : l'installateur de la suite logicielle (plus de 700Mo)

C'est avec cette *RC<sub>21</sub>* numéro 9 de la **4.17.1** que j'ai donc découvert plus en détails la suite logicielle.

21. *RC*: Release Candidate : version Candidate, une version expérimentale (interne) de celle qui sera livrée en fin de sprint

## 2. Validation de Release :

Afin de vérifier les fonctionnalités « **standards** » qui ont le plus de chance de présenter une régression, j'ai donc été formé au processus de validation de Release.

Ce processus consiste à effectuer les *SC<sub>22</sub>* définis dans un plan de test (Test Plan).



*« On Test (Test Plan), une release Test (RC), pour que le Testeur (système), puissent Tester (exécuter) des Tests (sur les caractéristique), de composants à tester (DUT)*

Figure 23 : Le test à Mu-TEST

Pour cela, Mu-TEST utilise des outils spécifiques au développement (disponible sur le réseau local)

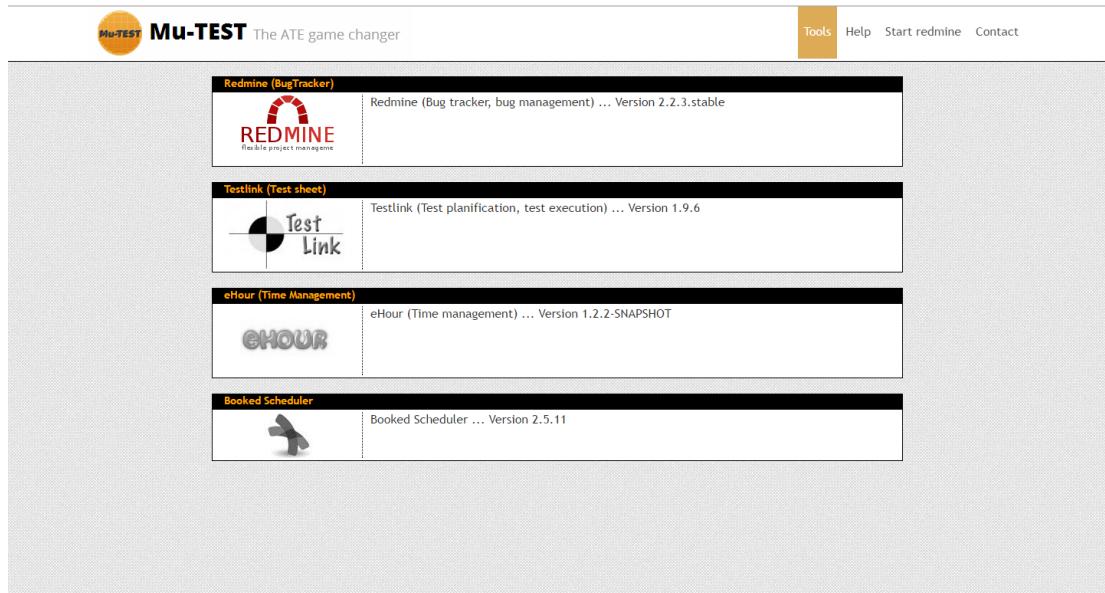
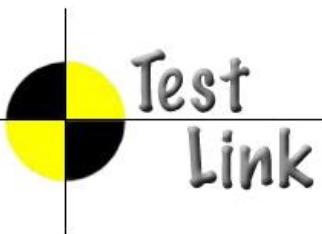


Figure 24 : La WorkStation comporte les outils au processus de Test (192.168.0.150)

22. *SC: Sanity Check : test de santé : on vérifie les fonctionnalités les plus importantes du système*

## TestLink : Test Plan



C'est avec **Test Link** que nous allons superviser le Test Plan à réaliser.

A chaque nouvelle Release, l'exécution du nouveau **Test Plan** doit être exécuté (partiellement ou dans son intégralité).

Ce Test Plan comporte des cas (Case) détaillés qu'il va falloir suivre. En fonction des résultats obtenus le Case sera soit : **PASS** (passer) ou **FAIL** (échouer)

Test Results on Build MuTest-setup-4.17.3-RC2-2016-06-03-08-30-05 | ?

Attention Please:  
Something is preventing connection to Bug Tracking System,  
and is going to create performance issues.  
Please contact your TestLink Administrator

Test plan notes  
Exhaustive test campaign

Build description

Print | Show complete execution history | Import XML Results |

Test Suite : New generation/ HSS - SC/ Golden 1/ Customer

Test Case ID mt-406 :: Version : 4  
DDR3  
No tester assigned

Last execution (any build) - Build : MuTest-setup-4.17.3-RC2-2016-05-23-17-12-13  
Date : 26/05/2016 14:25:23 - Tested by : jdutemps - Build : MuTest-setup-4.17.3-RC2-2016-05-23-17-12-13 - Status : Passed

Summary

Preconditions

Execution type : Manual

#	Step actions	Expected Results
1	Loadboard	<ul style="list-style-type: none"><li>Plug the DDR3 loadboard on the Golden 1 tester</li><li>Do not forget the loadboard stiffener</li></ul>
2	Test program	<ul style="list-style-type: none"><li>In MuTool load MT41J128M8.mtx test program</li><li>Folder D:\sw\dev\validation\ddr3_800mb\MT41J128M8\</li><li>Click on Validate</li></ul> <ul style="list-style-type: none"><li>DDR3 program loads in MuTool</li><li>No error/warning in Validate</li></ul>
3	MuTool	<ul style="list-style-type: none"><li>On Run tab select: Golden 1 IP address</li><li>Select package DDR3_800mb</li><li>Press on Load button</li><li>Select flow Flow_Test_demo</li><li>Press Start</li></ul> <ul style="list-style-type: none"><li>Load successful</li><li>Flow status : PASS</li></ul>

Notes / Description

Empty if it was PASSED  
Add information if it was FAILED

Result

Not Run  
 Passed  
 Failed  
 Blocked

Save execution  
Save and move to next

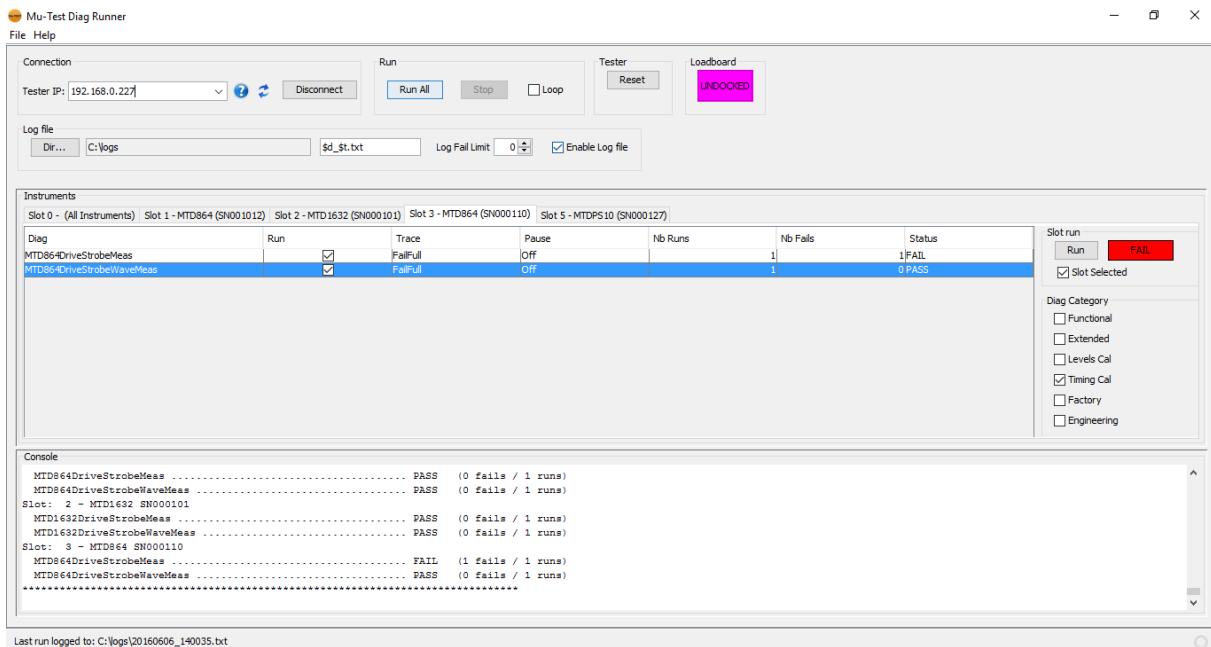
Important Notice: Once a Result is updated from 'Not Run' to another value, you cannot set it back to 'Not Run'. You can still set the Result to any other value.

Figure 25 : Le Test Case d'un composant - Une mémoire DDR3

De nombreux tests Case nécessitent des manipulations de composants, LoadBoard, instruments. Ceux-ci sont exécutés dans en **Laboratoire** dans des conditions anti-statiques (**Labo**) afin de prévenir les erreurs.

Si le Test est **PASS** et que cela semble cohérent alors tout est parfait.

Cependant, si après avoir suivi la procédure, il s'avère que le résultat obtenu est un **FAIL** :



**Figure 26 :** Echec d'une calibration

Il est alors nécessaire de comprendre les raisons de ce **FAIL**, pour cela direction les *Logs* générés.

```

7453
7454 Slot: 3 - MTD864 SN000110
7455 MTD864LevelsCalCheck ..... PASS (0 fails / 1 runs)
7456 MTD864StaticDclTest ..... PASS (0 fails / 1 runs)
7457 MTD864BasicPmupp ..... FAIL (1 fails / 1 runs)
7458 MTD864InternalMemories ..... PASS (0 fails / 1 runs)
7459 MTD864SequencerMemory ..... PASS (0 fails / 1 runs)
7460 MTD864PatternMemory ..... PASS (0 fails / 1 runs)
7461 MTD864BasicDriveAndStrobe ..... PASS (0 fails / 1 runs)
7462 MTD864BasicFtest ..... PASS (0 fails / 1 runs)
7463 MTD864ExtendedCapture ..... PASS (0 fails / 1 runs)
7464 | MTD864LevelsCal ..... FAIL (1 fails / 1 runs)
7465 | MTD864LevelsAccuracy ..... FAIL (1 fails / 1 runs)

```

**Figure 27 :** Aperçu synthèse d'un fichier Log généré par **MuDiagnostique**

Un Log peut comprendre de nombreuses lignes (exemple ci-dessus 7 465 lignes), c'est pour cela que dans un premier temps je vais consulter le résumé.

D'après une analyse rapide de ce Log, Il semblerait que certaines calibrations (ici **MuDiagnostique**) soit **FAIL** sur l'instrument **M-864** (Ref : SN000110) dans le Slot 3.

23. *Log*: « registre » : un Log contient des informations écrites par un logiciel. Lors de l'exécution d'une séquence test, il est possible d'activer l'enregistrement des données dans un fichier .txt/.log (fichier texte). Le Log contient alors des détails sur l'exécution des différents tests, ce qui permet de comprendre l'échec.

En consultant en détails ces différentes calibrations, je m'aperçois d'un point commun :

PMUPPIM RANGE_A_50MA Accuracy (Spec: +/- (100uA + 0.5%))										
Slot	Ctlr	Pin	Description	Expected	Lo Limit	Actual	Hi Limit	Error	%Limits	Result
3	1	29	PMUPPIM (RANGE_A_50MA)	-15mA	-15.175mA	-5.6358mA	-14.825mA	9.3642mA	<----->	**FAIL**
3	1	29	PMUPPIM (RANGE_A_50MA)	-12.5mA	-12.663mA	-5.6415mA	-12.338mA	6.8585mA	<----->	**FAIL**
3	1	29	PMUPPIM (RANGE_A_50MA)	-7.5mA	-7.6375mA	-5.6339mA	-7.3625mA	1.8661mA	<----->	**FAIL**
3	1	29	PMUPPIM (RANGE_A_50MA)	-2.5mA	-2.6125mA	-5.6358mA	-2.3875mA	-3.1358mA	<----->	**FAIL**
3	1	29	PMUPPIM (RANGE_A_50MA)	2.5mA	2.3875mA	-5.632mA	2.6125mA	-8.132mA	<----->	**FAIL**
3	1	29	PMUPPIM (RANGE_A_50MA)	7.5mA	7.3625mA	-5.6262mA	7.6375mA	-13.126mA	<----->	**FAIL**
3	1	29	PMUPPIM (RANGE_A_50MA)	12.5mA	12.338mA	-5.6301mA	12.663mA	-18.13mA	<----->	**FAIL**
3	1	29	PMUPPIM (RANGE_A_50MA)	15mA	14.825mA	-5.6339mA	15.175mA	-20.634mA	<----->	**FAIL**

8 fails / 512 tests

Figure 28 : Détail Log -calibration PMUPP

La Pin 29 semble être à l'origine de cela. Cette fois-ci cela n'est donc peut-être pas dû à une régression du Software mais plutôt à un problème Hardware (Pin endommagé ?)

C'est alors qu'intervient un second outil de la WorkStation : **RedMine**.

## RedMine : Bug Tracker

**RedMine**, est la plateforme recensant tous les problèmes connus (Software / Hardware / Client). En recherchant des informations dans la partie Hardware, je m'aperçois que j'ai affaire à une Anomaly référencée ouverte. C'est donc quelque chose de connu, je ne vais donc pas la signaler mais me renseigner si cela est toujours d'actualité.

Dans le cas contraire, j'aurais signalé une Anomaly et en aurais discuté avec l'équipe en question afin d'en définir l'état de priorité.

The screenshot shows a RedMine issue page for "Anomaly #1001". The title is "[LABO][D864][SN000110] Defaut de calibration pin 61". It was added by Pierre GYOMLAI 4 months ago. The status is New, priority is Normal, assignee is Pierre GYOMLAI, and the start date is 01/21/2016. The description section contains a code snippet from PMUPPMV.Pin[1, 29]: cal values (gain = -729.915672; offset = 19608236.527) out of range. The notes section states: La pin 61 correspond au composant MAXIM9979 en U65 (790\_0000\_00).

Status:	New	Start date:	01/21/2016
Priority:	Normal	Due date:	
Assignee:	Pierre GYOMLAI	% Done:	0%
Category:	-	Fixed in Version:	
Target version:	-	Non-compliance:	No
Steps to Reproduce:		Cause analysis:	
Target Customer:	MUTEST		
Found in Version:	4.17 BETA		

**Description**  
PB de calibration de la mesure PMUPP sur la pin 61 de la carte sn000110.

```
PMUPPMV.Pin[1, 29]: cal values (gain = -729.915672; offset = 19608236.527) out of range
PMUPPMI (RANGE_E_20UA).Pin[1, 29]: cal values (gain = 3346.519149; offset = -89816703.708) out of range
PMUPPMI (RANGE_D_20UA).Pin[1, 29]: cal values (gain = 2002.488889; offset = -53733416.414) out of range
PMUPPMI (RANGE_C_200UA).Pin[1, 29]: cal values (gain = 4971.696552; offset = -133443840.042) out of range
PMUPPMI (RANGE_B_2MA).Pin[1, 29]: cal values (gain = 14915.089655; offset = -400423944.876) out of range
PMUPPMI (RANGE_A_50MA).Pin[1, 29]: cal values (gain = 2448.976842; offset = -65734099.979) out of range
```

**Subtasks**  
**Related issues**

Figure 29 : Anomaly #1001 correspondant à notre FAIL

### 3. Post Procédure

A la fin du **Plan Test**, j'ai pris l'habitude de communiquer une synthèse référençant les **FAILS** rencontrés. Cela permet d'avoir une trace détaillée et de tenir informées les différentes équipes sur l'état de la Release.

MT	Name	Teste	Status	Comment	Date	TEST
Initialisation						
518	MuToolInstal		PASS		RC10	209
266	TortoiseHg		PASS		RC10	PASS
Golden 1						
518	Gateway 5		NOT RUN	No system (PASS 4.17.2 RC)		91
266	Load FPGAs	227	PASS		RC10	0,435407
340	Verification Diags	227	FAIL	validation\DiagGold1 need to be update	RC10	FAIL
435	DCC Cal	227	PASS		RC10	21
456	Level Cal	227	FAIL	Slot 3 Anomaly #1001	RC10	0,100478
429	Timing Cal	227	PASS		RC9	NOT RUN
270	S25 Hirex	227	PASS		RC10	97
271	MT29 Hirex	227	FAIL	Pattern Match Error	RC9	
272	MT47	227	PASS		RC9	
273	Hct163_Fmax	227	PASS		RC9	
274	MCP23017	227	PASS		RC9	
424	Hct163_Quad	227	PASS		RC10	
297	HW Check	227	PASS		RC9	
379	IS42	227	FAIL		RC9	
380	IS43	227+228	FAIL	Continuity (socket Issues A3) Anomaly #1135	RC9	
382	Hct163_Fmax1632	227	PASS		RC9	
381	HW Check1632	227	PASS		RC9	

Figure 31 : Aperçu Rendu du Test Plan complet de la 4.17.3RC10

Au cours de mon stage, j'ai également dû rédiger certains Tests Case.

Ayant apporté de nouvelles fonctionnalités Software au cours de mon activité, il a fallu également réaliser différents Tests Case afin de vérifier leurs fonctionnements dans leur intégralité.

De plus, j'ai été poussé à exécuter des tests en dehors du Plan Test. Des tests pour lesquels il n'existe aucun cas permettant de vérifier le fonctionnement car cette situation n'a pas été envisagée.

Cela a permis dans certains cas de rajouter de la sécurité quant à la robustesse de certaines fonctionnalités, ou encore mieux de remédier à certains fonctionnements qui ont été oubliés (en particulier pour des cas précis).

#### Reported issues (9)

#	Project	Tracker	Subject
1137	Software	Anomaly	[Test] Duration Test 1632 : Test Number (New)
1136	Software	Anomaly	[Test] Duration Test : Failed (New)
1135	Software	Anomaly	[Test] : IS43 - Timeout Vox (New)
1123	Software	Anomaly	[MuTool] Production Mode : Flow loaded (New)
1121	Software	Anomaly	[MuTool] Hct163_Fmax\Hct163_4x.mtx : Loop (Resolved)
1120	Software	Anomaly	[MuTool] First Validation : Burning Test (Resolved)
1119	Software	Anomaly	[MuTool] Warning : M-GTW6_Instrument (New)
1122	Software	Anomaly	[MuTool] Production Mode : Datalog (New)
1116	Software	Anomaly	[MuProdLauncher] UNUSABLE : Due to CTRL0 FPGA Rev mismatch (Resolved)

[View all issues](#)

Figure 31 : Des bugs que j'ai reportés

## L. Projet

### ● Introduction

En plus de participer à l'activité de maintenance, j'ai mené un projet répondant à un besoin de l'entreprise : la réalisation **d'un système de Tag** dans l'Outil **MuTool**.

Ce projet constitue la partie centrale de mon stage.

### 1. Besoin :

Suite à l'utilisation d'un testeur de Mu-TEST, un client a reproché un **manque d'ergonomie du panneau de Test** (Test panel) comportant les tests à exécuter.

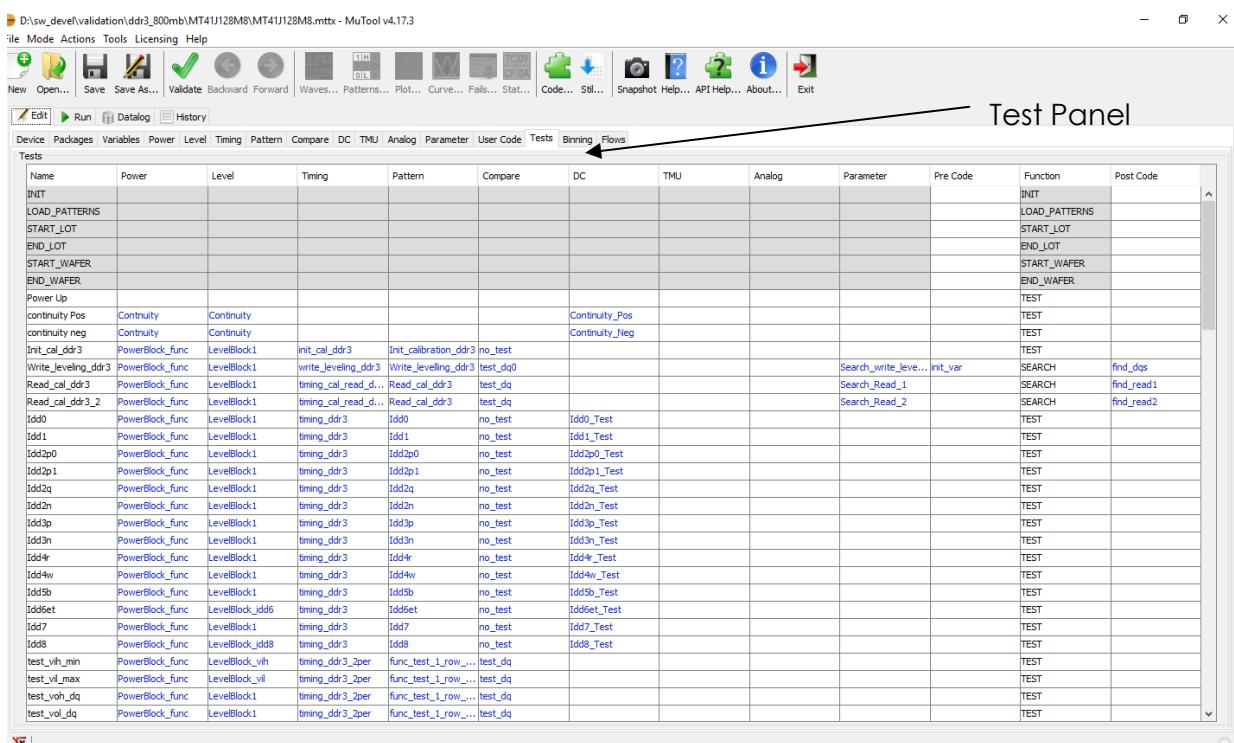


Figure 32 : Le Test Panel d'un programme d'une mémoire DDR3

En effet, un programme de test peut comporter énormément de tests différents. Il peut être difficile de les distinguer car certains n'ont qu'une unique variable qui est différente.

Afin de répondre à ce manque de visibilité et de satisfaire le client, il a été décidé de **réaliser un système de « Tag »**. N'étant pas une tâche prioritaire, celle-ci me fut confiée et fut prévue pour intégrer la futur 4.18.

## 2. Solution : La fonctionnalité de Tag

L'objectif de la fonctionnalité de Tag est de changer uniquement l'aperçu graphique d'un item (élément) d'une liste si celui-ci comporte un Tag affilié.

Le Tag correspondra à un groupe, par exemple en **jaune** les tests de continuité.

Le test en question prendrait alors la couleur du Tag (une couleur non tape-à-l'œil, afin de ne pas surcharger le panel), cela reste une indication visuelle.

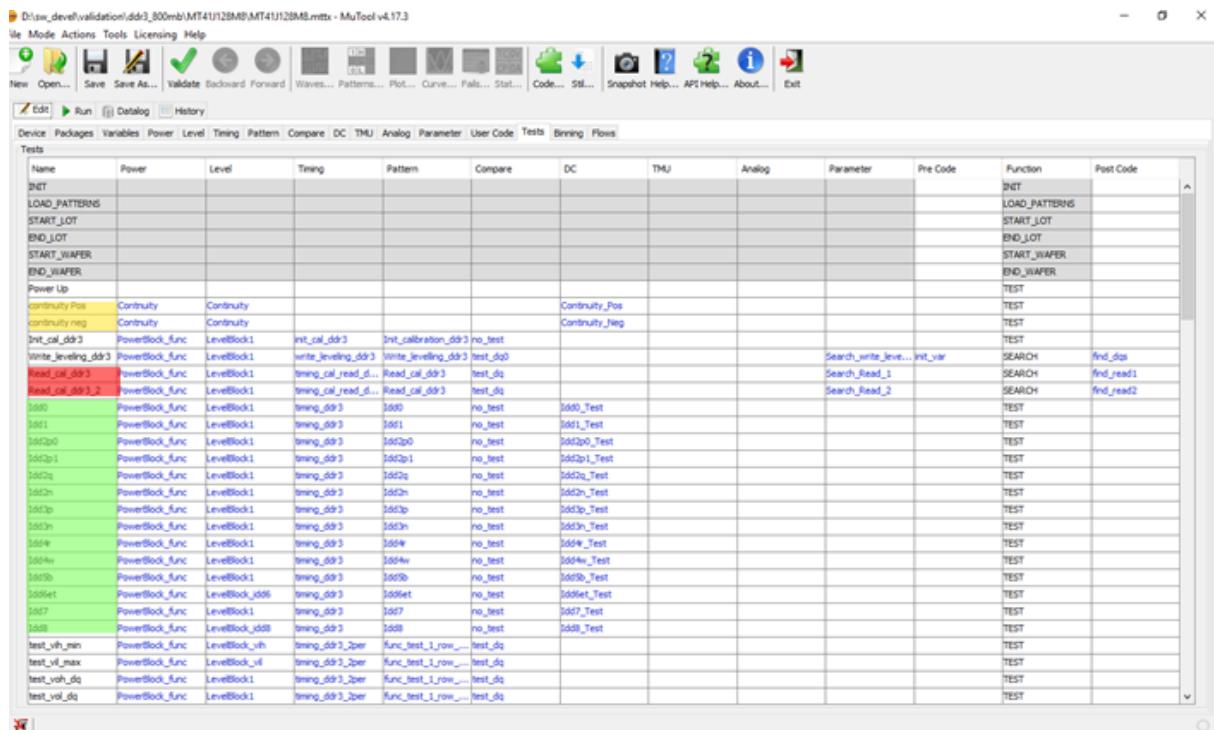


Figure 33 : Rendu souhaité

Dans un premier temps, cette fonctionnalité sera exclusive au Test Panel. Cependant, elle sera ensuite étendue à d'autres Panels. Il faut donc bien penser à développer l'outil de Tag comme une fonctionnalité modulable de Panel en Panel.

En plus du rendu graphique, dans la même optique d'amélioration de visibilité de l'interface, un système de filtre affichant uniquement les tests ayant une liste de Tags sélectionnés sera mis en place.

- Training :

La suite logicielle graphique étant développée en *Java*<sup>24</sup>, j'ai donc dû, dans un premier temps, apprendre les bases de ce langage. En effet, lors de ma formation, j'ai appris des langages de programmation tels le C, C++, C#, Python, SQL, PHP mais pas le Java

Cependant le fonctionnement de Java ne m'est pas complètement inconnu. Il possède de nombreux points de similitudes avec le C#. Langage que j'ai étudié au cours de ma formation afin de réaliser un projet : **PacMan XNA**



Pour prendre en main l'architecture de la suite logicielle de Mu-Test ainsi que l'*IDE*<sup>25</sup> **NetBeans** (*Outils Dev A11-A12*) j'ai donc commencé par un petit projet que m'a confié mon tuteur : la **AboutBoxGeneric**.

Dans un premier temps, suite à la sortie d'un nouveau logiciel (**MuProdLauncher**), j'ai tout d'abord pris en main la modification d'une interface graphique via le *Design*<sup>26</sup> (Java Swing) sur NetBeans afin d'intégrer la génération d'une boîte de dialogue (Popup) contenant les informations du logiciel, la AboutBox (A Propos).



Figure 34 : Design - AboutBox

Outre le Design, j'ai utilisé des méthodes de l'**architecture de MuTool** afin de récupérer diverses informations tels : le numéro de version, les notes de version afin de modifier des informations du Design. De plus, j'ai modifié des logiciels afin d'incorporer la Release utilisée dans la barre de titre.

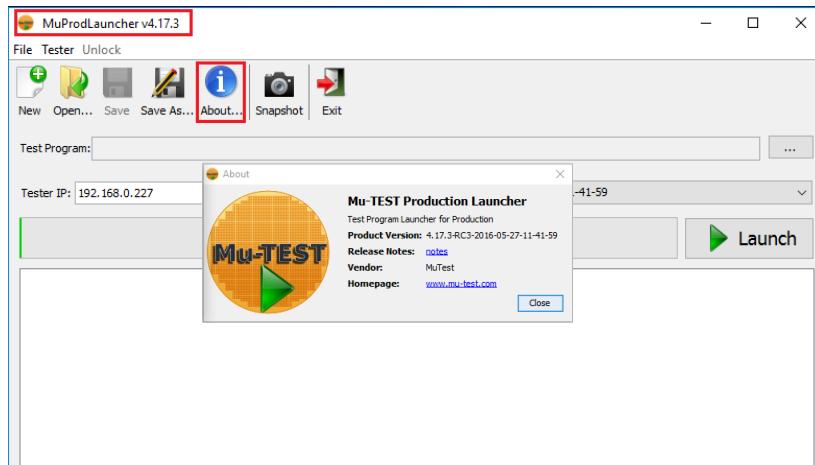


Figure 35 : MuProdLauncher - AboutBox

Afin de m'aider à la conception j'ai pu m'appuyer sur les **AboutBox** déjà présentes qui m'ont permis de mieux comprendre le fonctionnement de Java et de l'architecture.

24. *Java*: langage de programmation orienté objet, connu pour sa portabilité sur différents systèmes d'exploitation (IOS / Linux / Windows) ainsi que sa documentation très fournit.

25. *IDE* : Integrated Development Environment : environnement de Développement : ensemble d'outils pour augmenter la productivité d'un développeur (ex : Visual Studio, Eclipse, NetBeans)

26. *Design* : Modification du visuel avec des outils graphiques de l'*IDE*

Suite à cela, mon tuteur m'a proposé de créer une méthode générique **AboutBoxGeneric**, qui sera appelée dans les différents logiciels afin de générer la AboutBox avec les paramètres voulus. Ce besoin vient du fait, que certains programmes de la suite ne présentaient pas le même Design (taille, informations logo)

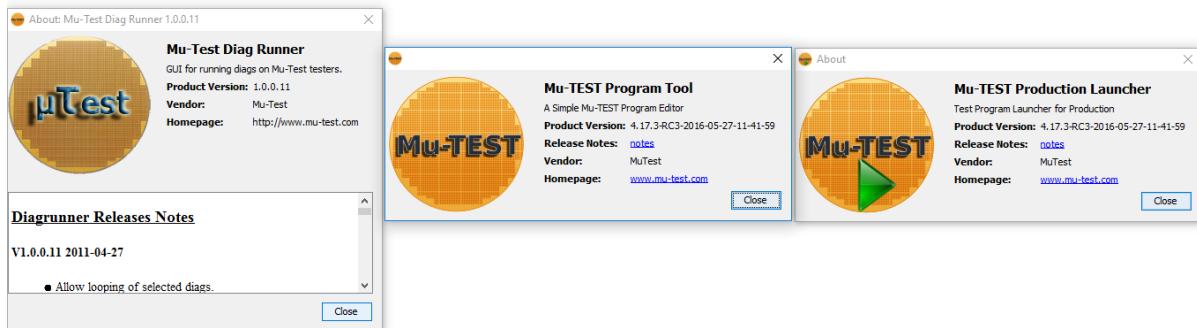


Figure 36 : Incohérence des AboutBox

```
* @author Vincent Berthet
*/
public class MuToolGenericAboutBox extends JDialog {

    /**
     * creator
     *
     * @param parent parent Frame
     * @param icon_path
     * @param title
     * @param description
     */
    public MuToolGenericAboutBox(Frame parent, String icon_path, String title, String description) {
        super(parent);
        initComponents();

        Map<TextAttribute, Object> underlinedBlueFont = new Hashtable<>();
        underlinedBlueFont.put(TextAttribute.UNDERLINE, TextAttribute.UNDERLINE_ON);
        underlinedBlueFont.put(TextAttribute.FOREGROUND, Color.BLUE);

        Font font = appNotesLabel.getFont();
        appNotesLabel.setFont(font.deriveFont(underlinedBlueFont));
        appHomepageLabel.setFont(font.deriveFont(underlinedBlueFont));

        imageLabel.setIcon(new javax.swing.ImageIcon(getClass().getResource(icon_path)));
    }
}
```

Figure 29 : Une partie de la méthode qui sera utilisé par les logiciels

```
@Action
public void showAboutBox() {
    if (aboutBox == null) {
        JFrame mainFrame = MuToolApp.getApplication().getMainFrame();
        aboutBox = new MuToolGenericAboutBox(mainFrame, "Icon path", "Title", "Description");
        aboutBox.setLocationRelativeTo(mainFrame);
    }
    MuToolApp.getApplication().show(aboutBox);
}
```

Figure 36 : Exemple - Utilisation de la méthode dans un logiciel

Le développement de l'**AboutBoxGeneric**, m'a permis de comprendre différents mécanismes du Java, de NetBeans (l'IDE) mais surtout le fonctionnement de l'architecture de la suite logicielle de Mu-TEST et les outils de développement tel le *Versioning*<sup>27</sup> avec l'outil Mercurial (*Outils Devs*)

27. *Versioning* : versionnage : désigne le mécanisme qui consiste à conserver la version d'une entité logicielle quelconque. Cela permet de modifier simultanément un programme par plusieurs personnes et de bénéficier des nouvelles modifications si elles sont partagées.

- Tags :

### 1. Base : Création de la Class Tag

Afin de pouvoir réaliser cette fonctionnalité, il va être nécessaire de créer une *Class*<sup>28</sup>, la **class Tag**.

Il est primordial que cette Class soit correctement construite car c'est elle qui est la base de la fonctionnalité.

Cette class permettra de définir le Tag. Elle se doit donc de contenir des variables tels :

- **le nom** : il permettra d'identifier le Tag facilement
- **une couleur** : afin de pouvoir différencier les différents Tags
- **un état** : true ou false, qui caractérise l'état du Tag (« Enabled »)
- **un commentaire** : il décrit plus précisément le Tag

Tag
String name
Color color
Boolean enabled
String comment

Figure 39 : UML simplifié

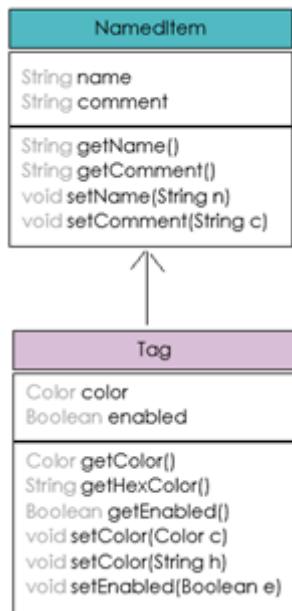


Figure 40 : UML intermédiaire

Cette Class comporte deux champs plutôt habituels que l'on retrouve dans de nombreuses Class: **name & comment**.

Or, le travail sur ces deux variables a déjà été réalisé et sert pour de nombreuses autres Class (tel la **Class Test**) qui en *héritent*<sup>29</sup>. Nous allons donc la faire hérité (extends) notre Class Tag d'une autre Class de l'architecture **MuTool** : la Class **NamedItem**.

Il va également être nécessaire de déclarer dans la Class Tag **des méthodes standards** telles les *Getters*<sup>30</sup> et les *Setter*<sup>31</sup> permettant la bonne interaction avec les variables de la Class.

Vous avez peut-être remarqué un Getter et un Setter particulier : **getHexColor()**, **setColor(String h)**. Ceux-ci sont nécessaires pour les imports/exports de la Class. On note que **setColor** est une méthode *surcharge*<sup>32</sup>.

En effet en héritant de la Class **NamedItem** on hérite également de ces *abstracts*<sup>33</sup>

28. *Class* : en programmation désigne un objet par des propriétés

29. *Héritage* : fait d'hériter des propriétés d'une autre Class.

30. *Getter* : méthode permettant l'accès des variables internes à une Class

31. *Setter* : méthode permettant la modification des variables internes à une Class

32. *Surcharge* : fait d'utiliser la même méthode mais avec des paramètres différents.

33. *Abstract* : méthode désignant l'utilisation d'une Class

Les abstracts de la Class **NamedItem** concernent notamment :

- ✓ l'import / export de la Class via un fichier .*CSV*<sup>34</sup>
- ✓ l'import / export du .*XML*<sup>35</sup> de la Class (XML qui n'est autre que le Programme de Test, les fichiers .mttx)
- ✓ des informations sur la modification de la Class (ligne de commande...)

En plus de l'abstract, j'ai dû écrire des fonctions communes afin de pouvoir récupérer par exemple la liste de tous les Tags existants dans toutes les Class de l'architecture.

Dans le but de modifier nos abstract pour notre Class, il nous faut *Override*<sup>36</sup> la méthode de l'abstract de la Class **NamedItem** afin que lors de l'enregistrement/ouverture du fichier (XML/CSV), la méthode appelée dans notre Class **Tag** exécute correctement l'import/export des Tags.

Or, **on ne peut écrire que des chaînes de caractère (String)** dans un fichier XML/CSV. La variable color appartenant à la Class Color (class interne à Java), il va donc nous falloir transcoder de la Class couleur à une chaîne de caractère transcodant une couleur (l'Hexadécimal).

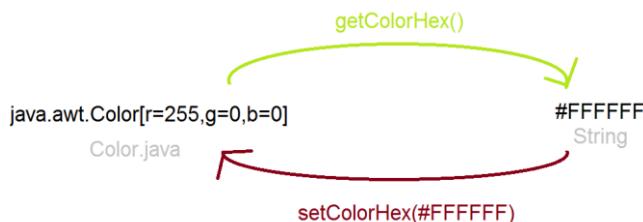


Figure 41 : Transcodage des Class Color/String

J'ai donc créé ces méthodes afin de pouvoir enregistrer/charger les variables de la Class Tag.

```
<tags>
  <tag id="3" name="Default Tag 1" color="#FF0000FF" enabled="1" comment="" />
  <tag id="4" name="Default Tag 2" color="#00FF00FF" enabled="0" comment="" />
  <tag id="5" name="Default Tag 3" color="#0000FFFF" enabled="1" comment="" />
</tags>
```

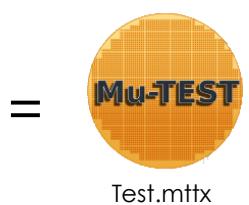


Figure 42 : Un fichier .mttx (XML), cela est un Programme de Test comportant des Tags



FF0000FF

Cela ne correspond pas un code Hexadécimal d'une couleur.

R G B ?

En effet, en avançant mon projet j'ai rajouté un champ supplémentaire à mes couleurs : **Alpha** (qui est la transparence) car j'ai eu besoin d'utiliser des couleurs transparentes.

34.*CSV* : format utilisé pour sauvegarder des données sous d'un tableau

35.*XML* : fichier texte utilisé pour la configuration

36.*Override* : surmonter : en Java permet de réécrire une méthode d'une Class



Et si on charge un ancien fichier qui ne connaît pas la fonctionnalité Tag ?

C'est là une force de l'**architecture MuTool**. En effet, lors de l'ouverture d'un programme .mttx (XML), on retrouve en entête du fichier la version dans laquelle il est sauvegardé. Si cette Release est antérieure à celle dans laquelle le programme est chargé, on appelle alors le **Migrator**.

```
<?xml version="1.0" encoding="UTF-8"?>
<! -- This file must not be edited, otherwise it cannot be loaded any more -->
<program name="MT41J128M8" version="4.16-BETA3">
```

Figure 43 : Entête d'un programme de Test MT41J128M8.mtxx (DDR3)

Le **Migrator** apporte les changements dans le XML jusqu'à celui de la Release utilisée.

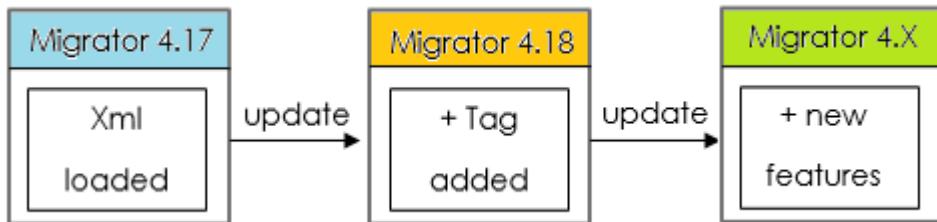


Figure 44 : Fonctionnement du Migrator

Notre **Migrator 4.18** devra intégrer la modification du XML des Tags (Figure 42).

Cela permettra alors de charger correctement un fichier de 4.17 depuis une 4.X. Celui-ci comportera alors les modifications des Tags (grâce à la migration de fichier avec le Migrator 4.18 et ensuite le Migrator 4.X)

Mon tuteur m'a ensuite demandé de réaliser un autre test. Cette fois ci, c'était un **Test Java**. J'ai donc écrit un fichier Test Java afin de tester les fonctionnalités décrites précédemment.

Ces fichiers de Test permettent depuis l'IDE (NetBeans) de vérifier le bon fonctionnement des Class lors du débogage.

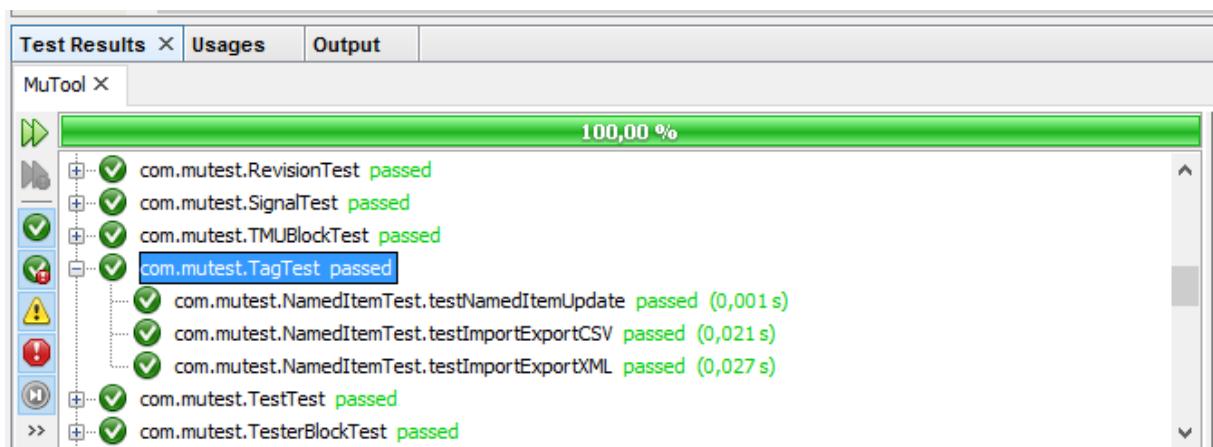


Figure 45 : Test de la Class Tag : Import/Export, création/modification de Tags

## 2. Affectation : Tag un Test

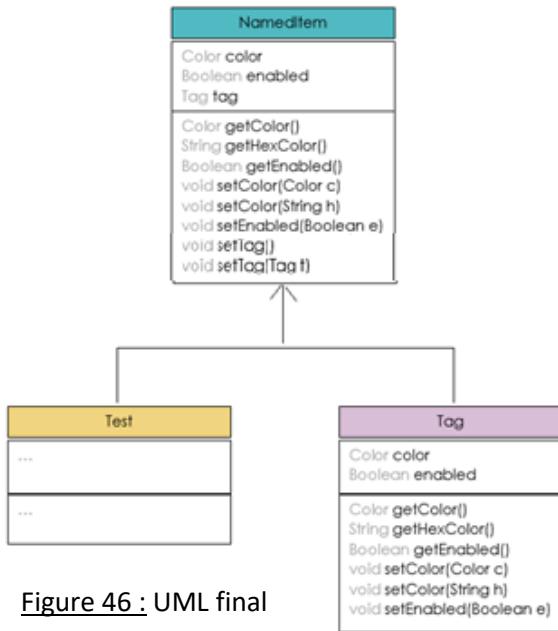


Figure 46 : UML final

Nous avons donc la base de notre fonctionnalité. Il nous faut désormais pourvoir **affilier une variable au Class que nous voulons Tagger**.

Or, la plupart des Class héritent de la Class **NamedItem** et en particulier les Class à qui l'on souhaite d'ajouter un Tag (Test pour commencer).

On va donc modifier **NamedItem** afin de pouvoir y insérer un champ comportant une String (chaîne de caractère) qui désignera le nom (variable name) du Tag cible.

Cette méthode permettant l'affiliation, s'appelle **setTag(Tag t)**.

Cependant il ne faut pas oublier de modifier l'abstract de la Class Test afin que cette variable soit prise en compte lors de l'import/export (XML & CSV) ainsi que le fichier permettant de vérifier le fonctionnement de la Class Test.

Nous disposons donc de la méthode permettant d'affecter correctement un Tag à un Test (et plus précisément à toute Class héritant de **NamedItem**).



Comment exécuter la méthode setTag(Tag t) ?

Pour cela j'ai ajouté un Menu au Popup Menu du Test Panel. C'est ainsi que nous allons Tagger nos items.

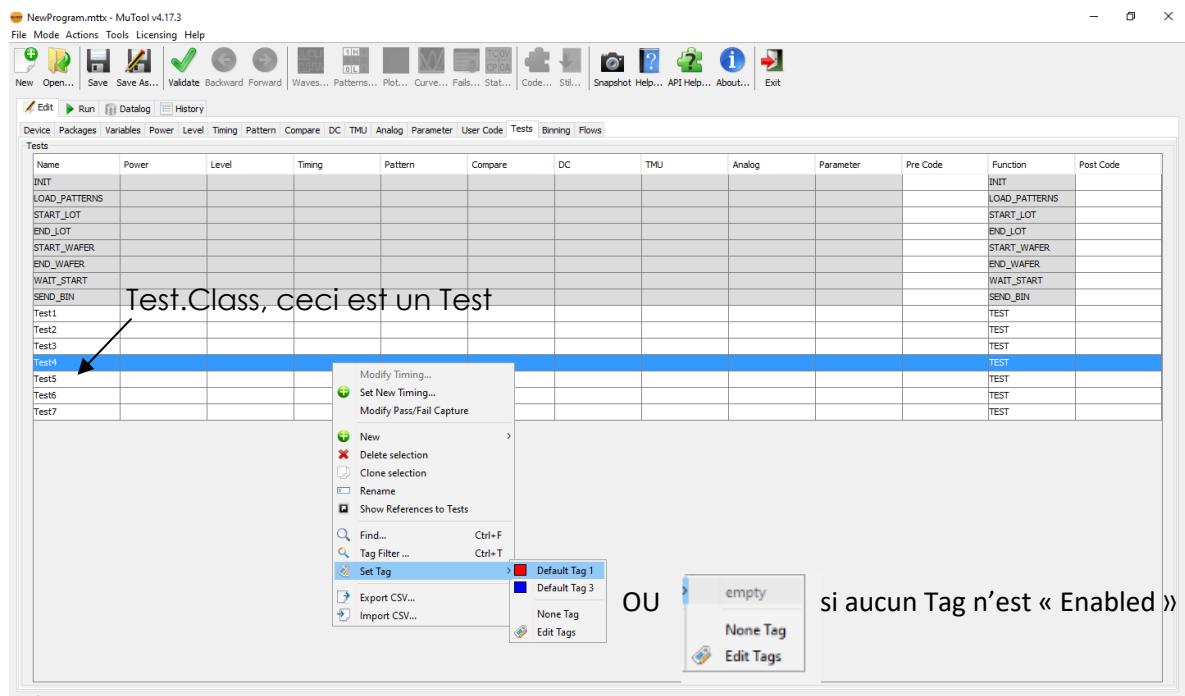


Figure 47 : Le Popup permettant de Tagger un item

La liste est une liste dynamique. Elle **est chargée à chaque nouvelle requête** au Popup Menu, elle contient uniquement les Tags « Enabled ». L'icône associé à chaque nom de Tag change en fonction de la couleur du Tag. J'ai dû créer spécialement une Class pour cela : **ColoredIcon**.

#### Au chargement de la liste :

- ✓ récupération du ou des Tests sélectionnés à Tagger (dans le cas contraire **le menu Set Tag...** est désactivé)
- ✓ récupération des Tags « Enabled »
- ✓ création d'une icône à la couleur du Tag (grâce à la précédente méthode **getColor()** ! )
- ✓ création d'une action **setTag(Tag t)** avec le Tag utilisé pour chaque item de la liste du menu.

Lors du clic sur un item de la liste du Popup Menu, on effectue notre action de Taggage de l'item.

Vous avez peut-être remarqué l'item « **None Tag** » dans la liste du Popup Menu. Celui-ci permet de supprimer le Tag affilié (toujours grâce aux méthodes de la Class **NamedItem**, ici c'est **setTag()** ).

Outre cette modification, il a fallu également mettre à jour une fonctionnalité du Test Panel appelée **Clone**.

Cette fonctionnalité permet **de dupliquer un Test avec des paramètres identiques** (sauf pour le « name » qui doit être impérativement unique,).

**Identiquement**, cela signifie donc que le Tag associé doit être lui aussi cloné. Voilà pourquoi il a fallu ajouter l'appel de la méthode **getTag()** et **setTag(Tag t)** dans l'action de clonage.

```
...
<tags>
    <tag id="3" name="Default Tag 1" color="#FF0000FF" enabled="1" comment="" />
    <tag id="4" name="Default Tag 2" color="#00FF00FF" enabled="0" comment=" /-
    <tag id="5" name="Default Tag 3" color="#0000FFFF" enabled="1" comment="" />
</tags>
...
<tests>
    <test id="18" name="INIT" function="INIT" comment="" testnum="0" />
    <test id="19" name="LOAD PATTERNS" function="LOAD PATTERNS" comment="" testnum="0" />
    <test id="20" name="START_LÖT" function="START_LÖT" comment="" testnum="0" />
    <test id="21" name="END_LÖT" function="END_LÖT" comment="" testnum="0" />
    <test id="22" name="START_WAFER" function="START_WAFER" comment="" testnum="0" />
    <test id="23" name="END_WAFER" function="END_WAFER" comment="" testnum="0" />
    <test id="24" name="WAIT_START" function="WAIT_START" comment="" testnum="0" />
    <test id="25" name="SEND_BIN" function="SEND_BIN" comment="" testnum="0" />
    <test id="111" name="Test1" function="TEST" comment="" tag="Default Tag 1" testnum="0" />
    <test id="112" name="Test2" function="TEST" comment="" tag="Default Tag 3" testnum="0" />
    <test id="113" name="Test3" function="TEST" comment="" tag="Default Tag 3" testnum="0" />
    <test id="114" name="Test4" function="TEST" comment="" tag="Default Tag 1" testnum="0" />
    <test id="115" name="Test5" function="TEST" comment="" testnum="0" />
    <test id="116" name="Test6" function="TEST" comment="" testnum="0" />
    <test id="117" name="Test7" function="TEST" comment="" testnum="0" />
    <test id="118" name="Test12" function="TEST" comment="" tag="Default Tag 1" testnum="0" />
    <test id="119" name="Test22" function="TEST" comment="" tag="Default Tag 3" testnum="0" />
    <test id="120" name="Test32" function="TEST" comment="" tag="Default Tag 3" testnum="0" />
    <test id="121" name="Test42" function="TEST" comment="" tag="Default Tag 1" testnum="0" />
    <test id="122" name="Test52" function="TEST" comment="" testnum="0" />
    <test id="123" name="Test62" function="TEST" comment="" testnum="0" />
    <test id="124" name="Test72" function="TEST" comment="" testnum="0" />
</tests>
...
```

Figure 48 : On a affilié des Tags à certains Tests, par ailleurs on a fait un clone des Test d'ID 111 à 117

### 3. Rendu : Tag Renderer



Et le rendu dans tout cela ?

Justement, à ce stade nos **items (éléments) de la class Test** possèdent uniquement un champ « Tag=myTag » comme désiré, mais le rendu est le même que initialement. Pour y remédier, il va falloir modifier l'affichage : le **Renderer** (rendu).

Au début de cette partie, je vous ai exprimé le rendu que l'on souhaitait obtenir ([B - 2. Solution : La fonctionnalité de Tag – P32](#)).

Il faut bien comprendre qu'ici, nous n'allons en **aucun cas modifier les données**. Ceci correspond **uniquement à de l'affichage**. Nous allons créer une nouvelle méthode d'affichage qui prendra les Tags en compte. Je l'ai appelée **MuTagRenderer**.

Ce Renderer spécifique fonctionne comme suit :

- ✓ sa méthode récupère en paramètre la Class utilisée de l'item en question (ici c'est la Class Test)
  - ✓ par défaut (Abstract d'un Renderer) celui-ci récupère les informations de la cellule (table, value, ligne, colonne ...)
  - ✓ si cela est possible, il récupérera la Class **NamedItem** (comportant les variables Name, Comment et Tag, [Figure 46](#)) de cet Item.
- Grâce au travail précédent **Test hérite d'un Tag affilié grâce à son héritage de la Class NamedItem.**
- ✓ on regarde ensuite si cet item possède un Tag et si celui-ci est « Enabled »
  - ✓ si c'est le cas :
    - on récupère alors le Tag (`getTag()`)
    - on affecte la couleur du Tag au Background (fond) de la cellule (`getColor()`)
    - on affecte en **Tooltip** le commentaire du Tag (`getComment()`), cela permet lors du pointage de la souris sur la cellule en question d'indiquer une instruction supplémentaire.
  - ✓ sinon : on ne change rien au Renderer (affichage par défaut)

Edit Tags			
Tags			
Enabled	Color	Name	Comment
true	java....	Default Tag 1	
false	java....	Default Tag 2	
true	java....	Default Tag 3	

Edit Tags			
Tags			
Enabled	Color	Name	Comment
<input checked="" type="checkbox"/>	red	Default Tag 1	
<input type="checkbox"/>	green	Default Tag 2	
<input checked="" type="checkbox"/>	blue	Default Tag 3	

Figure 49 : 2 Panels identiques sauf en affichage

A savoir que le Renderer est **appelé automatiquement à chaque changement de propriété**

Ici nous souhaitons seulement affecter la première colonne. Le **Renderer** sera donc appelé uniquement sur cette colonne (grâce à la méthode Java `setRenderer(MuTagRenderer(Test.class))` ).

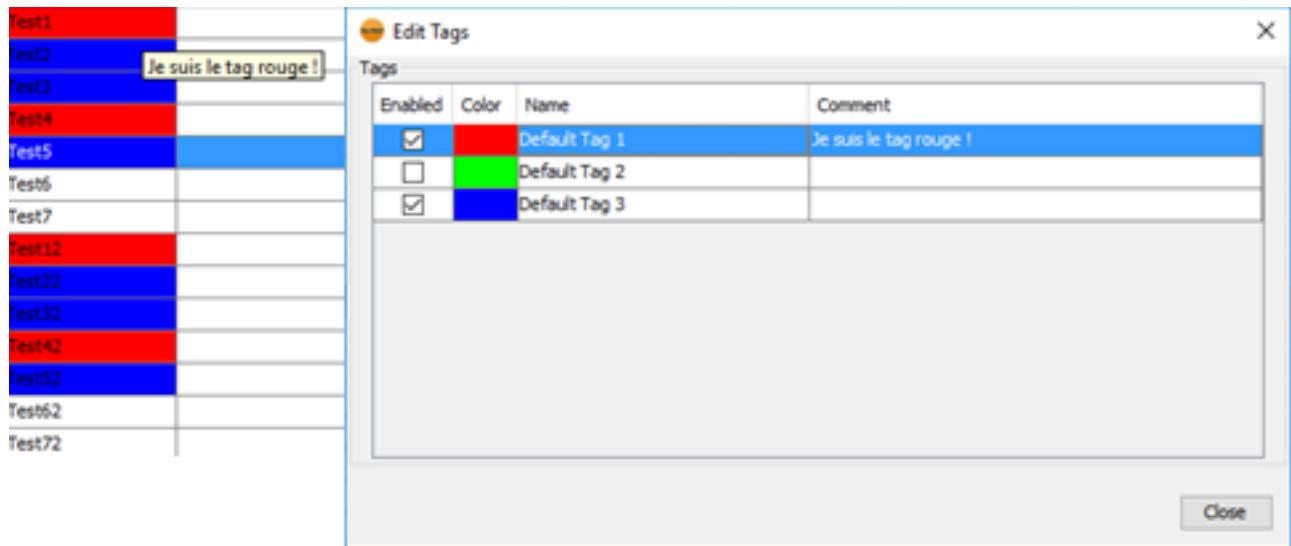


Figure 50 : MuTagRenderer sur la première colonne (les Tests sont taggés d'après Figure48)

On note que si un Test présente le Tag « Default Tag 2 » alors **MuTagRenderer** ne modifera pas son rendu (car ici désactivé)

## 4. Edition : Modifier les paramètres des Tags



Comment modifier les Tags ?

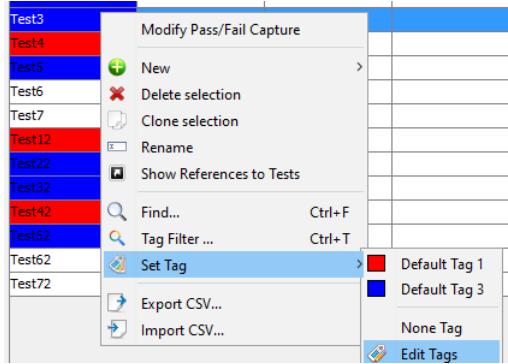


Figure 51 : Edit Tag Menu Item

Dans cette partie, je vais vous expliquer **comment j'ai modifié les paramètres de nos Tags**. Pour cela je vais devoir utiliser les **Setter** de la Class **Tag** (color et enabled) et ceux de la Class **NamedItem** (name et comment). Cependant, pour interagir avec ceux-là, il faut **une interface dédiée** !

Pour ne pas surcharger l'interface du Test Panel, pourquoi ne pas utiliser un Popup ? (tel la **ShowAboutBoxGeneric**). Ce Popup s'ouvrira lorsque l'on souhaitera éditer un ou plusieurs Tags.

On utilisera l'item nommé « **Edit Tags** » du Popup Menu du TestPanel» afin d'appeler l'outil d'édition de nos Tags.

L'action sur « **Edit Tags** » permettra de générer le Popup suivant :

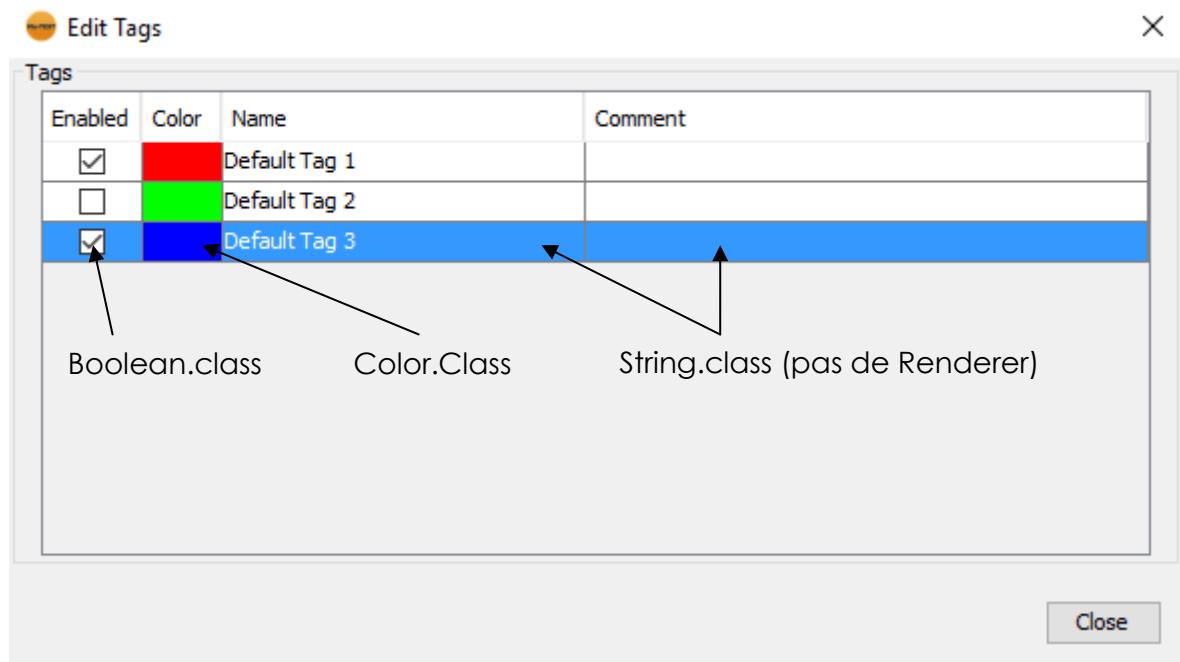


Figure 52 : Le Popup d'édition de Tag

On charge dans ce Popup la liste de tous les Tags qui existent. De la même manière que pour le Panel de Test, on utilise un nouveau Renderer afin d'afficher le rendu graphique souhaité :

**MuColorRenderer** (La Figure 49 illustre l'apport de **MuColorRenderer**).

De plus, j'ai ajouté une fonctionnalité qui **sélectionne automatiquement le Tag** du premier Test sélectionné lorsque le Popup est appelé. C'est une fonctionnalité simplement ergonomique.

L'édition de nos Tags doit permettre d'exercer sur notre liste de Tags, les actions suivantes :

- |   |  |
|---|--|
| <ul style="list-style-type: none"><li> Ajout</li><li> Suppression</li><li> Renommage</li><li> Changement de couleur</li><li>Activation / Désactivation</li><li> Utilisation (références)</li><li> Import / Export CSV</li></ul> | <ul style="list-style-type: none"><li>✓ D'un Tag unique (nom unique, comme pour les Test)</li><li>✓ Du Tag &amp; et de son utilisation</li><li>✓ Modifier le nom du Tag</li><li>✓ Modifier la couleur du Tag</li><li>✓ Activer ou Désactiver la visibilité du Tag sur les Panel</li><li>✓ Voir dans quel Test le Tag est utilisé</li><li>✓ Importer/ Exporter la liste de nos Tag correctement</li></ul> |
|---|--|

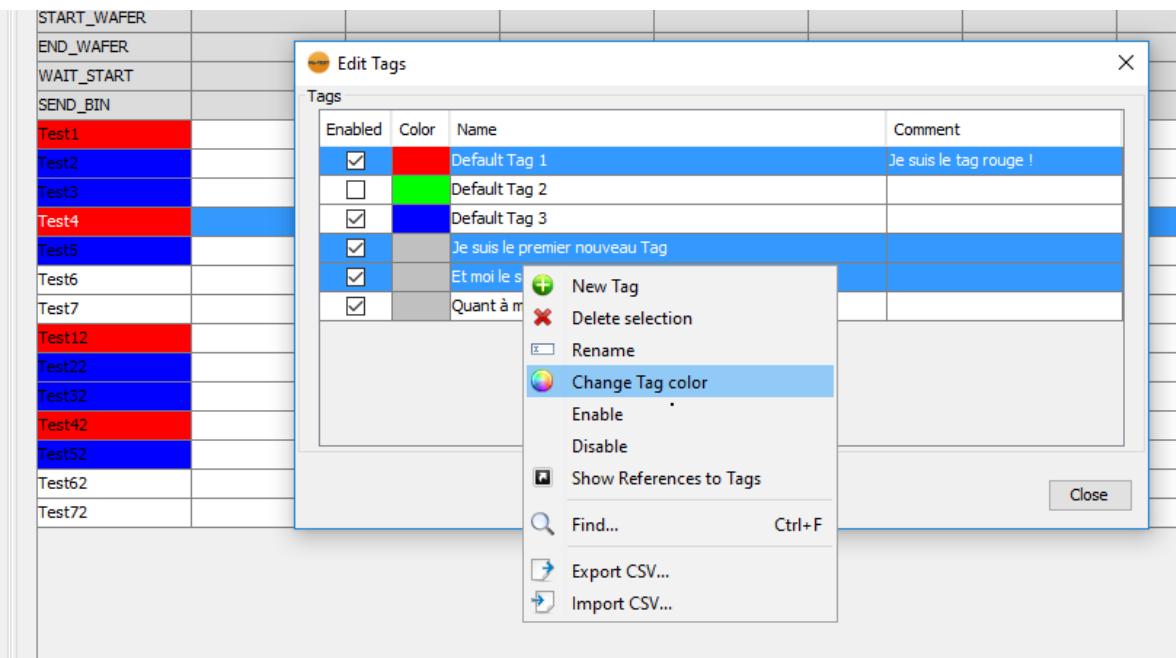


Figure 53 : Les actions sur le Popup d'édition des Tags

(Voir annexe *Edition pour aperçu de fonctionnement*)

Toutes ces actions **utilisent différents Getter et Setter** afin de pouvoir modifier correctement le programme. Il est essentiel que ces fonctionnalités marchent correctement car la stabilité du programme en dépend.

### Prenons l'exemple de la suppression :

Nous utiliserons un programme de Test expérimental simplifié.

Ce programme de Test est composé **uniquement** de Tags qui ont été définis par défaut à la création du programme .mttx (XML).

On a également ajouté à ce programme quelques Tests afin de pouvoir y vérifier notre fonctionnalité. Parmi ces Tests certains possèdent un Tag : « **Default Tag 1** » ou encore « **Default Tag 2** » voire pour certains aucun Tag « **""** » (Figure 48).

On peut représenter notre situation actuelle (ce que comporte MuTool) par le diagramme de la Figure 54.

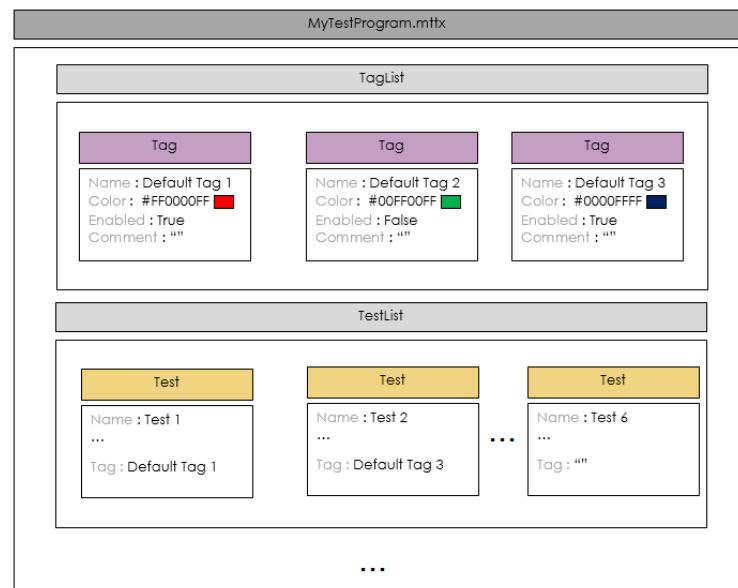


Figure 52 : Situation expérimentale - initiale



Supprimons le Tag « Default Tag 1 »

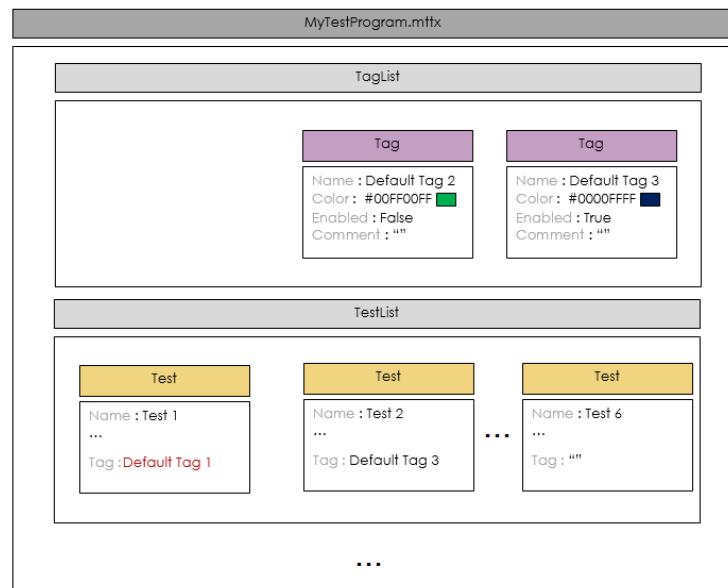


Figure 55 : Situation expérimentale - suppression

On se retrouve alors dans la configuration suivante :

- ✓ « Default Tag 1 » a été correctement supprimé de la TagList.
- ✓ « Test 1 », « Test 2 », ... « Test 3 » ont des Tag affiliés.

En effet le Tag a été correctement supprimé, cependant ce Tag était également référencé dans d'autres parties du Programme (ici « Test 1 »).

➔ Notre programme va rencontrer des problèmes lors de son chargement : **il ne sera pas chargé**

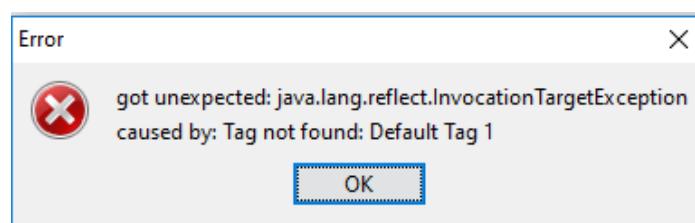


Figure 56 : Message d'erreur MuTool : lors du chargement d'un Program.mtx sans un Tag affilié à un Test(XML)

Dans ce cas précis, il va donc bien falloir en plus de supprimer le Tag, bien gérer la modification des Class comportant le Tag supprimé. Il est également important **de prévenir l'utilisateur** que sa suppression va impacter le reste du programme ([Voir annexe Edition](#)).

Cet exemple nous a permis de voir **qu'une simple erreur dans la modification d'une Class peut avoir de lourdes conséquences**. De plus, on constate que dans ce cas-ci, l'architecture de **MuTool** fonctionne très bien et que les erreurs Software sont directement remontées.

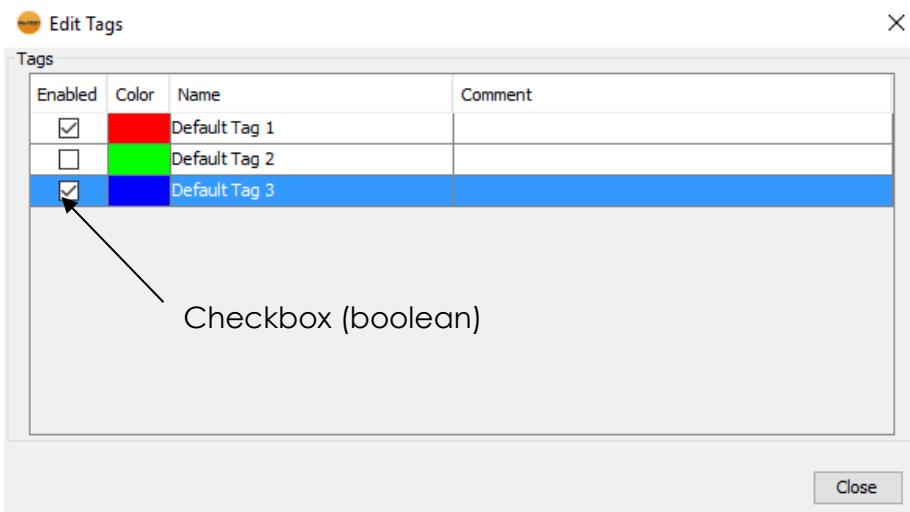


Pourquoi ne pas faire uniquement un bouton pour éditer les couleurs ?

En effet, c'est une très bonne idée. Cependant cette fonctionnalité présente une limite : **on ne modifie qu'un unique Tag** (le bouton sélectionne une seule ligne).

Précédemment le Popup a permis de modifier **une ou des couleurs** (en fonction des Tags sélectionnés). La fonctionnalité **d'utiliser la couleur tel un bouton** afin de pouvoir la modifier est très ergonomique ; nous allons incorporer celle-ci en complément de la précédente.

Tout d'abord je vous rappelle que le **Renderer permet uniquement de modifier le rendu visuel !**



Cela signifie donc que même si on voit des checkbox à la place de boolean (true/false), Il est **impossible de modifier leur statut**. Ceci n'est pas le rôle du Renderer, mais de l'**Editor**.

Nous allons donc devoir utiliser des Editors spécifiques qui lors de l'action déclencheront alors l'édition.

Figure 57: **MuTagDialog** avec des Renderers (et des Editors ?)

En ce qui concerne l'édition de boolean (représentée ici par les checkbox), il existe déjà un éditeur (**MuCheckEditor**) que nous allons donc Override pour l'utiliser sur la variable « enabled » de notre class Tag. En revanche il n'y en aucun existant pour éditer une couleur ; nous allons donc créer un éditeur dédié à cela : **MuColorEditor**.

### Fonctionnement du MuColorEditor :

- ✓ rendre les cellules de la colonne cliquables (transformation en bouton)
- ✓ ne pas avoir l'animation du bouton (conserver la couleur en fond de cellule)
- ✓ ouvrir l'outil en permettant la modification de la couleur

Boolean.Class Color.Class String.Class

Enabled	Color	Name
<input checked="" type="checkbox"/>		Default Tag 1

Figure 58 : La cellule rouge est un bouton

Vous pouvez remarquer que les colonnes de chaque tableau sont d'un unique type (Figure 58)

Le **MuColorEditor** sera donc exclusivement utilisé sur la colonne « Color ». Les autres disposeront soit d'un autre Editor (**MuCheckEditor**, **MuNameEditor** qui seront override pour avoir le fonctionnement souhaité) soit d'aucun si on ne veut pas les éditer.



Quel outil est utilisé afin de modifier la couleur de nos Tags ?

Dans ce cas, il ne s'agit pas uniquement d'une inversion de variable (MuCheckEditor – qui utilisera la méthode **setEnabled(Boolean b)** de Tag.Class).

J'ai donc procédé à la création d'un outil dédié à cela : **MuColorChooser**

### Fonctionnement du MuColorChooser :

- ✓ récupération en paramètre du ou des Tags à modifier
- ✓ sauvegarde de la couleur du premier Tag sélectionné
- ✓ chargement d'un panel prédéfini permettant de modifier la couleur
- ✓ la couleur par défaut de ce panel d'édition est initialisée à la couleur sauvegardée
- ✓ incorporation du panel d'édition à un nouveau Popup
- ✓ ce Popup dispose de trois actions :
  - **Confirm** : modification de la couleur de tous les Tags sélectionnés par la nouvelle couleur lors de la fermeture du Popup
  - **Reset** : réinitialisation de la couleur du panel d'édition à celle sauvegardée à son chargement
  - **Close** : fermeture du Popup

A la fin de la modification, il est important de rappeler le **Render** afin que celui-ci rafraîchisse l'affichage pour qu'il soit cohérent avec les nouvelles valeurs.

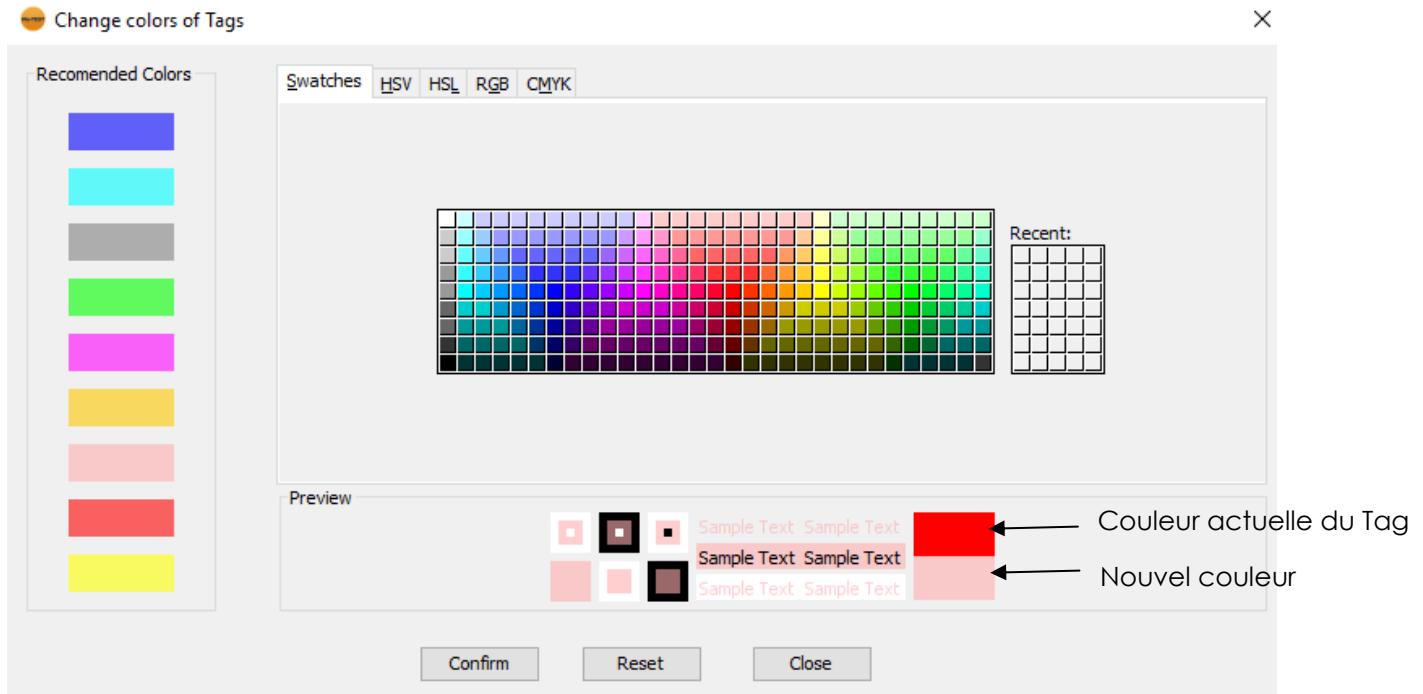


Figure 59 : Le Popup d'édition de couleur - MuColorChooser

Ce Popup est composé de trois différents panels :

- **Recomended Colors** : dans ce panel, on retrouve une **liste de couleurs recommandées par MuTest** à utiliser en tant que Tag. En effet, il est important qu'un Tag reste un rendu indicatif et non un voyant car il sert en tant que repère visuel. Ceci justifie la présence de couleur Recomended
- **Default Color Chooser** : celui-ci comporte le mélangeur de couleur par défaut de Java. Il est incorporé afin de permettre à l'utilisateur s'il le souhaite de modifier à son goût la couleur.
- **Action** : il comporte les trois actions de modifications décrites précédemment.

L'action de cliquer sur une couleur permet de saisir une nouvelle couleur dans le « Preview » (Prévisualisation) du **Color Chooser**. C'est cette couleur (de type `Color.Class`) qui sera affectée au Tag en sortie du Popup (via Confirm uniquement) grâce toujours à une méthode appartenant à `Tag.Class : setColor(Color c)`.

Vous pouvez noter que les **Recomended Colors** sont des couleurs « **plus transparentes** » que celles du **Color Chooser**. Ceci est dû au fait qu'en plus de modifier le code couleur avec le `Color Chooser`(Code couleur RGB), j'ai ajouté un paramètre du couleur : **Alpha** (la transparence).

Voici le paramètre Alpha que j'ai incorporé dans l'import/export du programme (Figure 34) !

(*Voir annexe **ColorChooser***)

## 5. Filtrage

Mon tuteur m'as proposé une nouvelle amélioration afin d'augmenter à nouveau l'ergonomie du Test Panel : réaliser **un système de filtrage des Tags**.

Ce filtrage permettrait de réunir plusieurs Tags au sein d'une liste de Tags (Filter List). Cette **Filter List** sera ensuite exploitée dans un premier temps dans le **Test Panel**. Elle permettra d'afficher uniquement dans le Panel **les Tests ayant un Tag à la liste**, et en conséquence de simplifier l'affichage du Panel.

Tout d'abord j'ai créé une interface permettant de réaliser cela, voici ses caractéristiques :

- ✓ inclusive dans tous les panels
- ✓ permettre de modifier la Tag List
- ✓ sauvegarder des Tag List
- ✓ permettre la navigation entre les différentes Tag List
- ✓ Comporter un raccourci pour l'édition des Tags.

Afin de pouvoir l'inclure dans chaque Panel, je vais devoir créer un outil spécifique, je l'ai nommé : **MuTagFilterPanel**.

Dans l'optique de ne pas surcharger les panels, celui-ci sera caché par défaut. On utilisera le Popup Menu du Panel afin de l'afficher. Le **MuTagFilterPanel** s'ajoutera alors en bas du Panel en question.

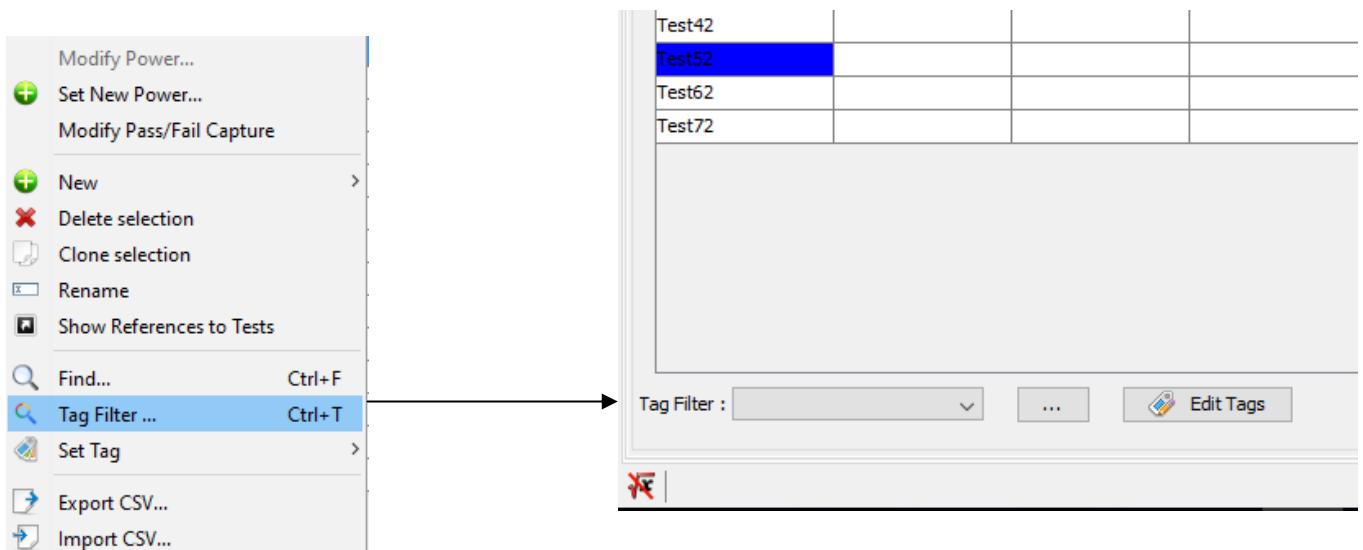
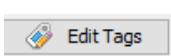


Figure 60 : MuTagFilterPanel

Cet outil dispose d'une boite de sélection. **Celle-ci contiendra les différentes Tag List du Programme** et affichera le nom de celle en cours. La boite de sélection nous permettra donc de choisir le filtre désiré à appliquer. En fonction de ce filtre, le Test Panel par exemple **affichera uniquement les Test composé d'un Tag de la TagList**.



Le bouton suivant fonctionnera de la même manière « Edit Tag » du Popup Menu lié au Test Panel.



Quant à celui-ci, il ouvre une nouvelle fenêtre Popup pour la gestion de la Tag List sélectionnée dans la boite de sélection.

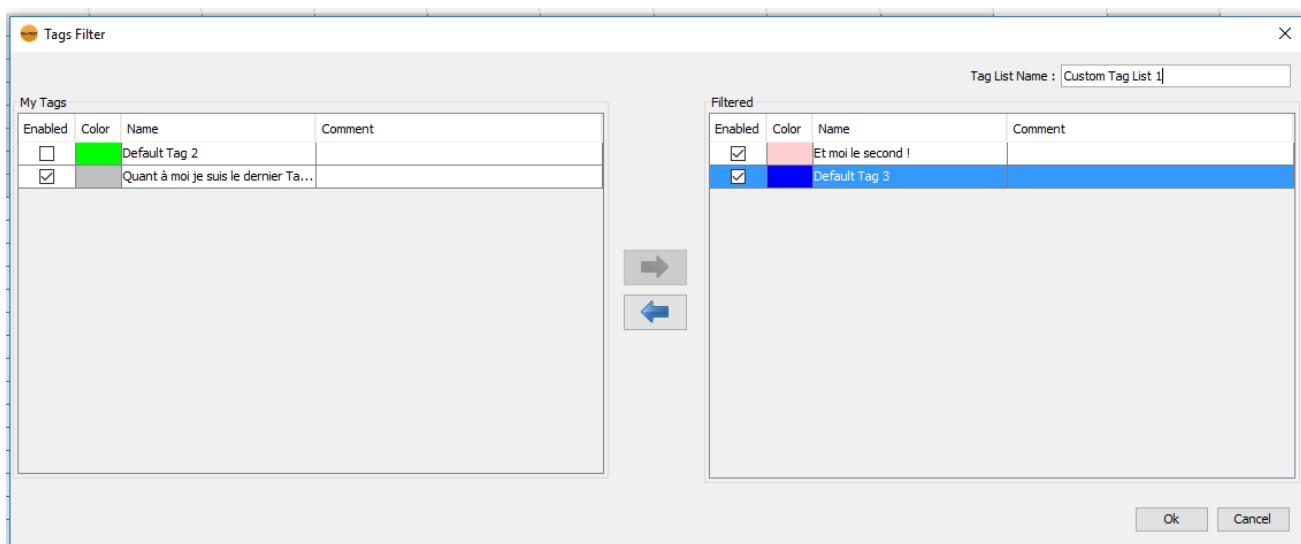


Figure 61 : MuTagFilterDialog

Dans cette exemple, les Tags « Et moi le second ! » et « Default Tag 3 » constituent la Tag List nommée « Custom Tag List 1 ».

Si la Tag List est vide, alors on supprime la List et la boite de sélection ne comporte pas le nom d'une TagList.

Les différents **Renderers** et **Editors** utilisés pour le Panel « Edit Tags » sont également présents dans les panels qui composent le **MuTagFilterDialog**.

([Filter](#))

➔ Durant le mois de stage qu'il me reste à effectuer, je vais donc finir le filtrage des Tags.

De plus, suite aux différentes démonstrations de mon projet lors des réunions de fin de Sprint, des améliorations m'ont été proposées. Je vais notamment devoir adapter le **MuColorChooser** pour élargir les nuances de la palette des Recommended Colors.

Une très bonne idée a également été soumise : pourvoir charger uniquement les Tests de la TagList dans le testeur (au lieu de tous les Tests), ce qui faciliterait certaines manipulations.

# VI. Conclusion

---

Au cours de ces trois mois de stage au sein de **Mu-TEST**, j'ai pu découvrir un environnement professionnel proche de ma formation. Outre mes précédentes expériences professionnelles, celle-ci est la plus longue et la plus enrichissante en tous points grâce au lien avec ma formation du diplôme universitaire en génie électrique et informatique industriel.

En effet, même si le milieu de l'ATE m'étais totalement inconnu pour moi à mes débuts le 4 Avril. J'ai pu dans un premiers temps apprendre rapidement les objectifs et les fonctionnements de l'ATE grâce aux équipes de Mu-TEST. Les modules, tels celui sur les FPGAs ou bien ceux d'études et réalisations lors de ma formation m'ont permis d'appréhender beaucoup plus facilement le milieu du test de composants. C'est ainsi que j'ai appris à connaître plus en détails le fonctionnement des composants électroniques et de comprendre les projets en cours dans l'entreprise.

De plus, j'ai découvert dans un second temps découvrir le développement software. Un domaine que je voulais approfondir dans le monde professionnel car cela représente pour moi la perspective d'étude que je pense mener.

Bien que n'ayant jamais étudié le Java au cours de ma formation j'ai me suis adapté rapidement à ce langage. C'est dans cette optique de découverte que mon tuteur m'a dirigé à mes débuts afin que je puisse comprendre le fonctionnement de Java, l'architecture software et les outils utilisés. Je pense notamment au versioning qui était totalement nouveau pour moi.

Par la suite, mon tuteur m'a confié un projet à mener en indépendance totale : la réalisation d'un système de Tag. Au cours de mon projet, j'ai pris le temps nécessaire à mettre au point les fonctionnalités définies durant les sprints.

En responsable de ce projet, j'ai présenté l'avancée de ce projet aux différentes équipes de sprint en sprint. Cela m'a permis de suivre des recommandations supplémentaires à celle de mon tuteur quant à la forme et à la méthode pour faire avancer le projet dans le bon sens.

L'ensemble du stage au sein de Mu-TEST m'a permis de parfaire des connaissances sur le monde professionnelle, le développement software et la communication. Cela m'a consolidé dans mon idée de poursuite d'étude de l'informatique en école d'ingénieur après mon année en DUETI où je vais étudier le Game Development.

Pour conclure ce rapport, je tiens à remercier à nouveau les personnes ayant contribués à la réussite de mon stage. Stage qui n'est pas encore terminé dans mon cas, ce qui va me permettre de compléter encore plus mon projet.

## VII. Annexes

---

- Système :



Figure A1 : M5S



Figure A2 : M21S



Figure A3 : Un testeur concurrent moins puissant pour un encombrement bien supérieur

- Labo :

Le Labo dispose d'équipements antis-statiques afin de préserver les composants tels le sol, des talonnettes, des blouses.



Figure A4 : Blouse anti statique

La capacité dissipatrice de la blouse nous permet d'éviter de décharger l'électricité statique de notre corps dans nos instruments ou dans les composants que nous testons. Le Nega-Stat utilisé dans les blouses permet de décharger dans l'air les charges électrostatiques accumulées (voir « effet couronne » sur Wikipédia si vous désirez en savoir plus.)



Figure A5 : Talonnette et chausson anti statique

Ces chaussures nous permettent d'éviter de générer des charges électrostatiques avec le frottement des chaussures de ville. Elles sont en effet beaucoup plus conductrices que les chaussures que nous portons au quotidien.

L'électricité statique, si elle n'est pas gérée dans un environnement de pointe comme le nôtre, peut rapidement causer la destruction de matériel électronique. Les dégâts causés peuvent rapidement atteindre des sommes colossales car le défaut causé n'est, la majorité du temps, pas visible à l'oeil nu.

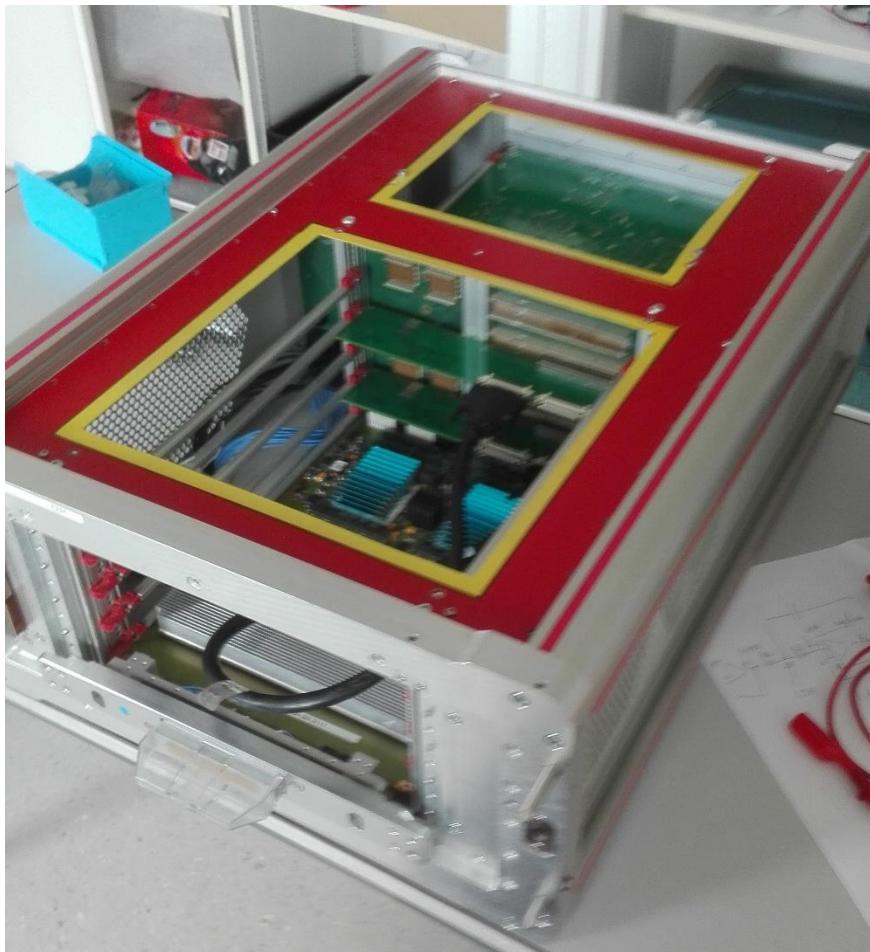


Figure A6 : Modification des instruments d'une M5S

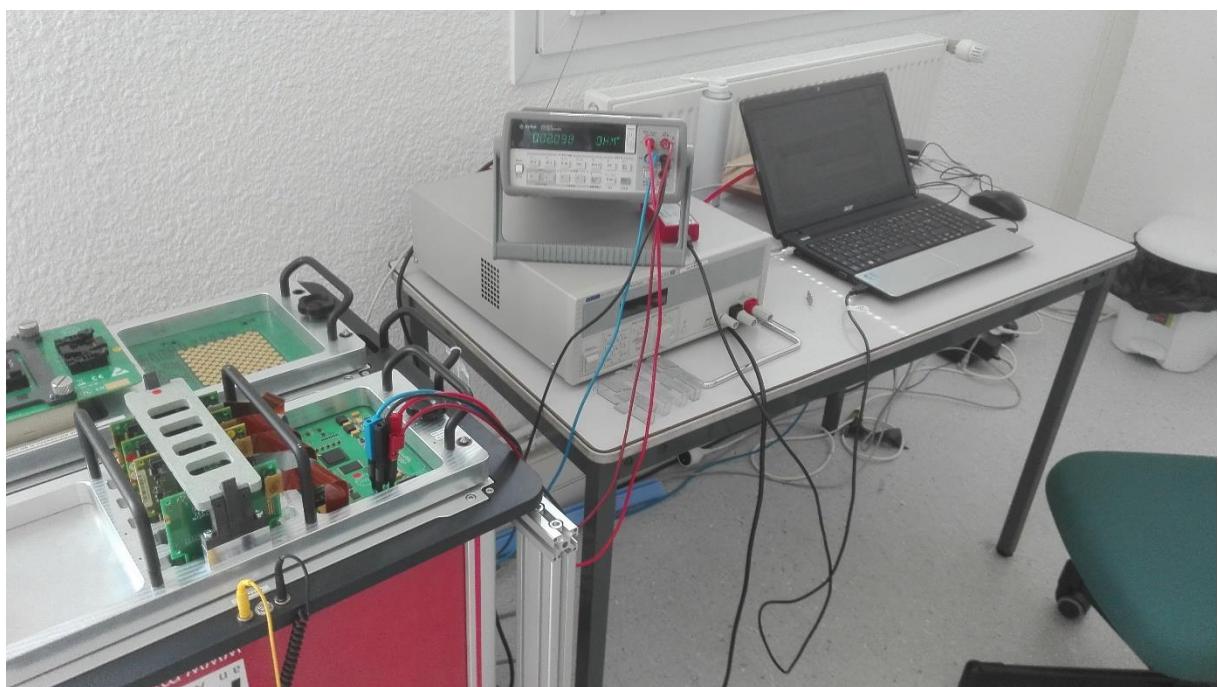


Figure A7 : Réalisation d'une Calibration manuel



Figura A8 : Binbox

La Binbox est un outil physique développé par Mu-TEST. Son but est faciliter l'utilisation de MuTool en production. Cela à un opérateur d'interagir et de surveiller rapidement et facilement avec l'exécution d'un programme de test.

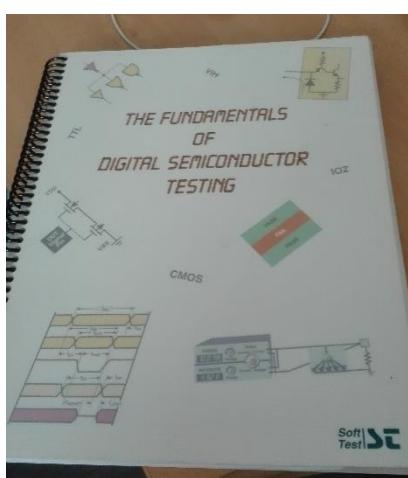
Elle dispose de 2 bouttons

- Start : Pour démarrer un programme chargé
- Abort : Pour annuler un programme en cours.

Et de 3 voyants :

- PASS
- IN PROGRESS
- FAIL

Un handler ou un prober peut également être utilisé avec.



Ce livre m'as permis de comprendre plus en détails le fonctionnement des composants et les tests à effectuer.

Figure A10 : Les fondamentaux sur le test de semi-conducteur numérique

## • Outils développement :

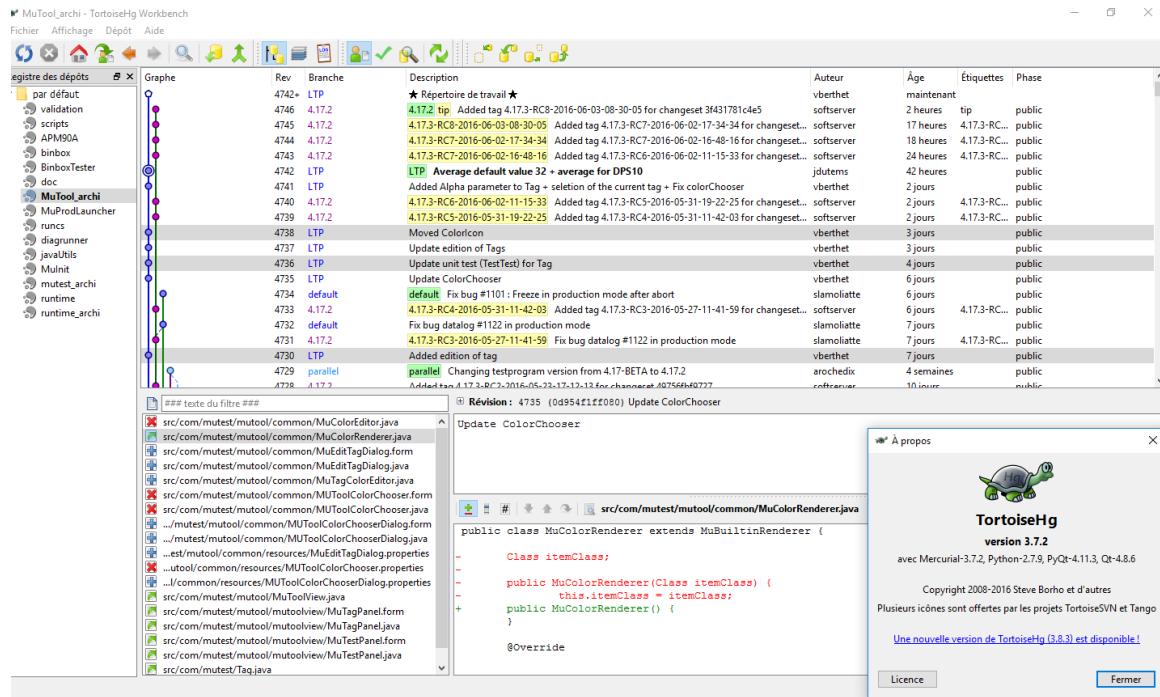


Figura A11 : Mercurial

C'est l'outil utilisé pour le Versionning. Le versionning permet à plusieurs personnes de développer en même temps.

```
ca "D:\sw-devel\scripts\pull.bat"

added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\levelsviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\levelsviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\linktableviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\memorypoolviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\mtd864dccviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\mtmixvviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\mtsn\
D:\sw-devel\viewers\mtto\
D:\sw-devel\viewers\rwt\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\sequencermemviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\statusviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\testereepromviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\testerviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\timingcalviewer\
added 4 changesets with 4 changes to 1 files
D:\sw-devel\viewers\viewerlib\
added 10 changesets with 15 changes to 6 files
Appuyez sur une touche pour continuer...
```

Figura A12 : Pull.bat , un script permettant de récupérer les révisions entrantes

Figura A13 : NetBeans – Code

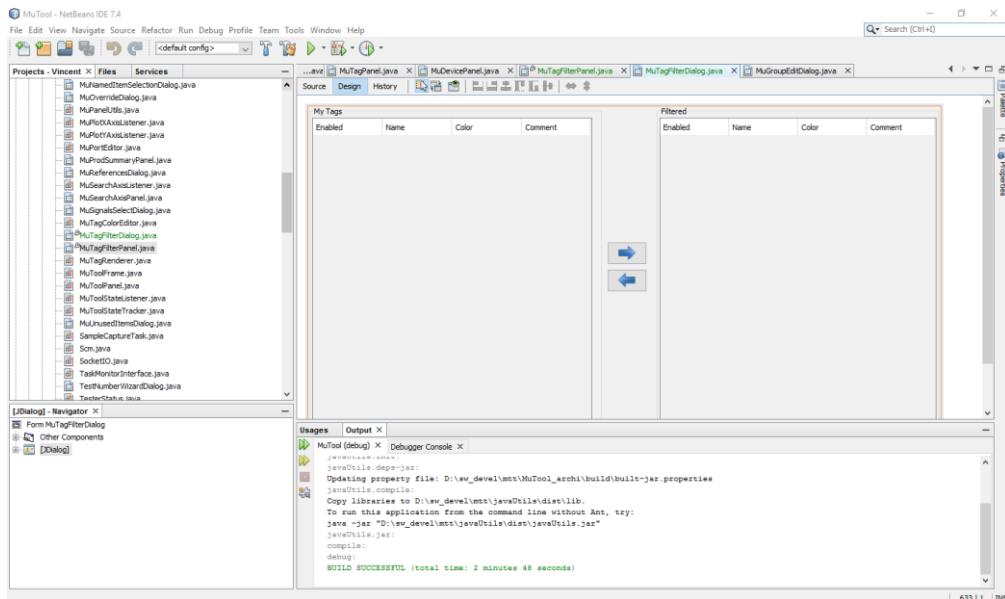


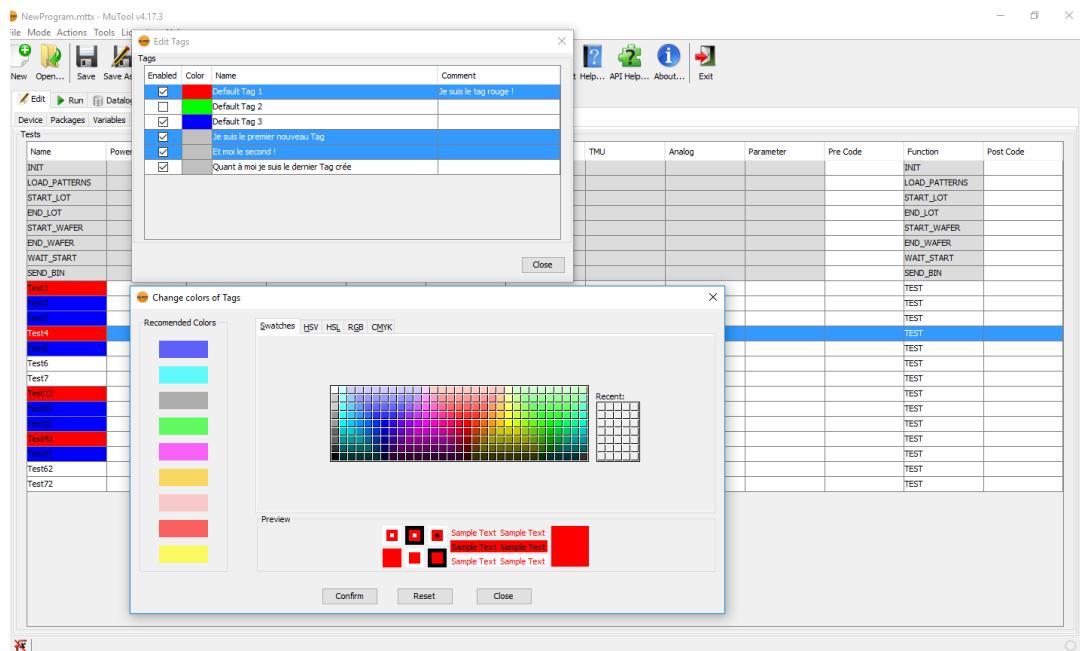
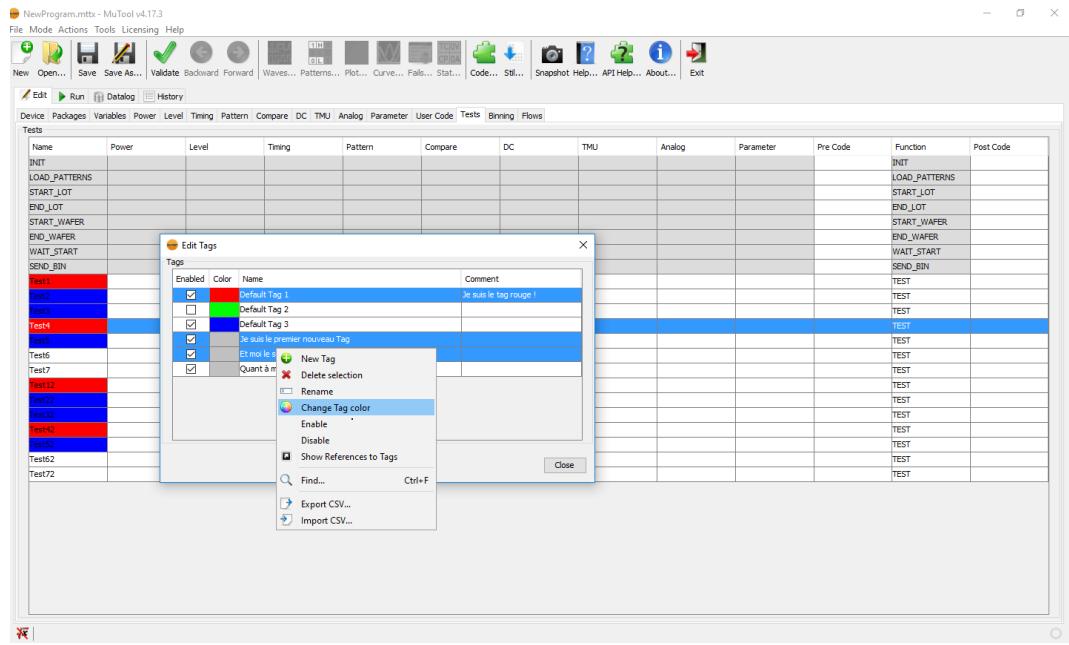
Figura A14 : NetBeans – Design

NetBeans est une plateforme de développement open source développée par Oracle. Il est utilisé par l'entreprise pour la création et la compilation du code créé, que ce soit pour la plateforme logicielle ou pour l'User code.

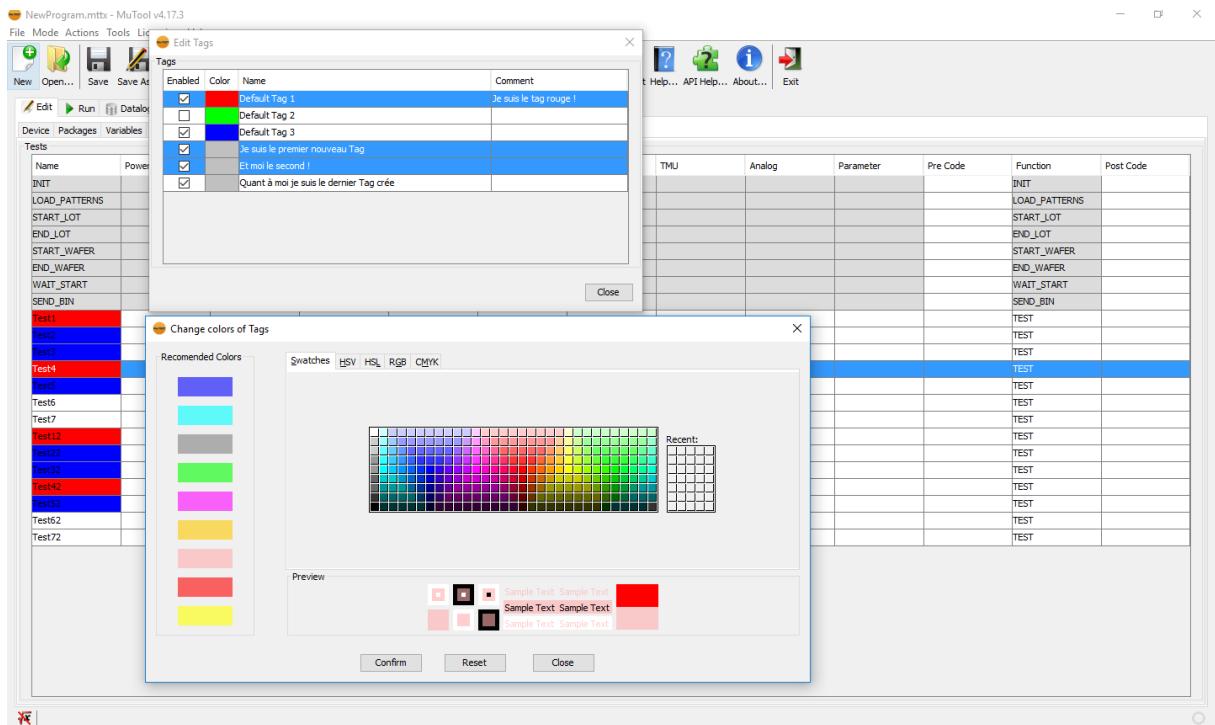
L'utilisation de cet outil de travail commun permet de minimiser, au niveau du développement logiciel, les problèmes venant de l'utilisation de multiples versions de plateformes de développement. Cet outil nous permet à la fois de coder en Java (pour la partie graphique) et en C++ (pour le back-end et pour l'User Code) ce qui en fait un outil très utile dans toutes les équipes R&D. Avec Eclipse, c'est l'une des interfaces de développement les plus utilisées dans l'industrie.

- Édition :

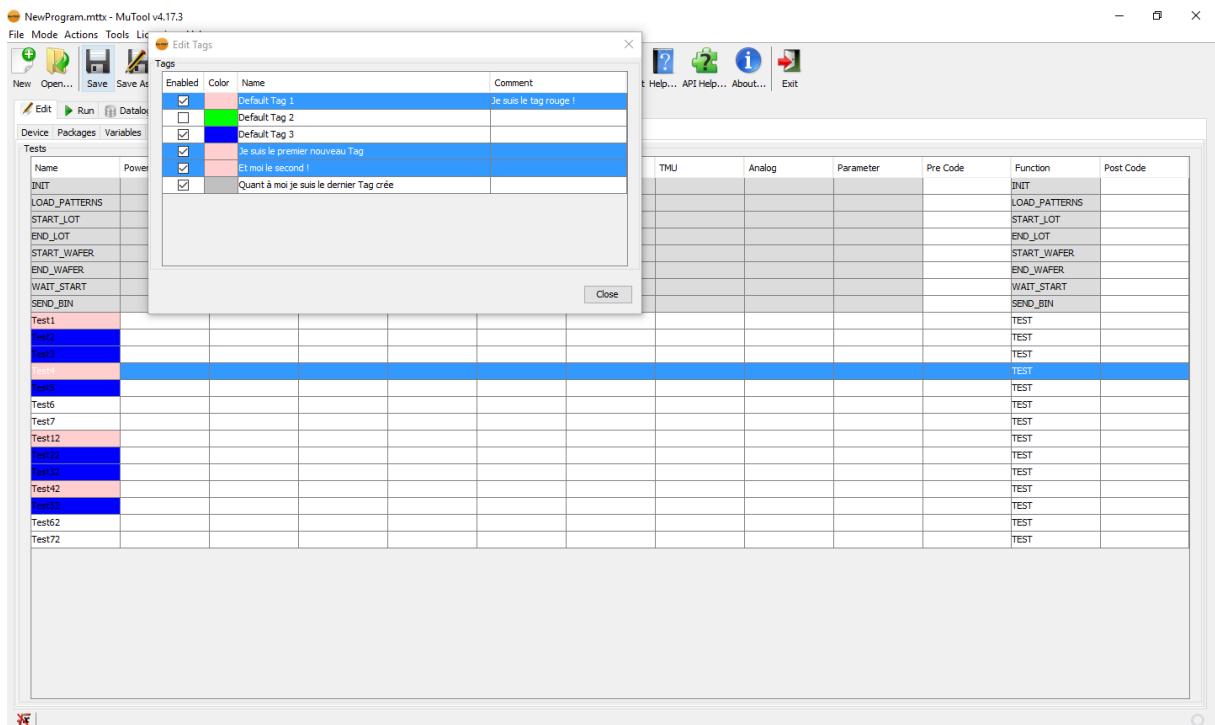
Modification de plusieurs couleurs :



La couleur rouge est prise, cela correspond à la couleur du premier Tag sélectionné.

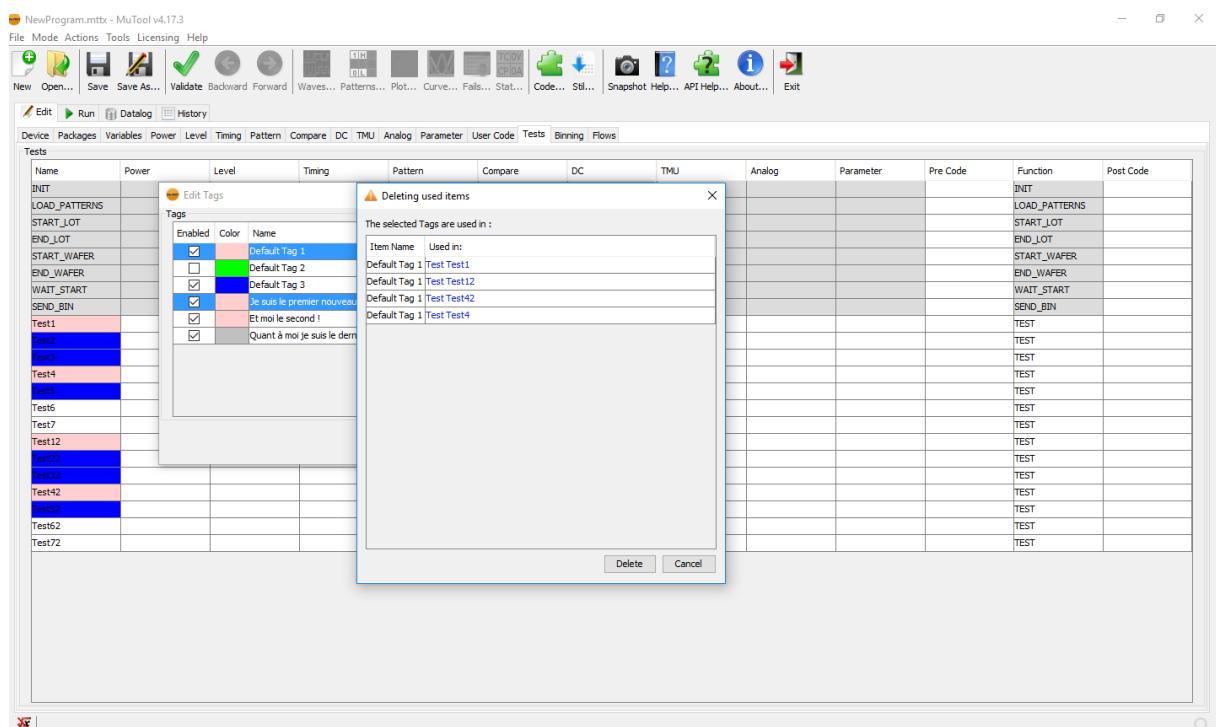
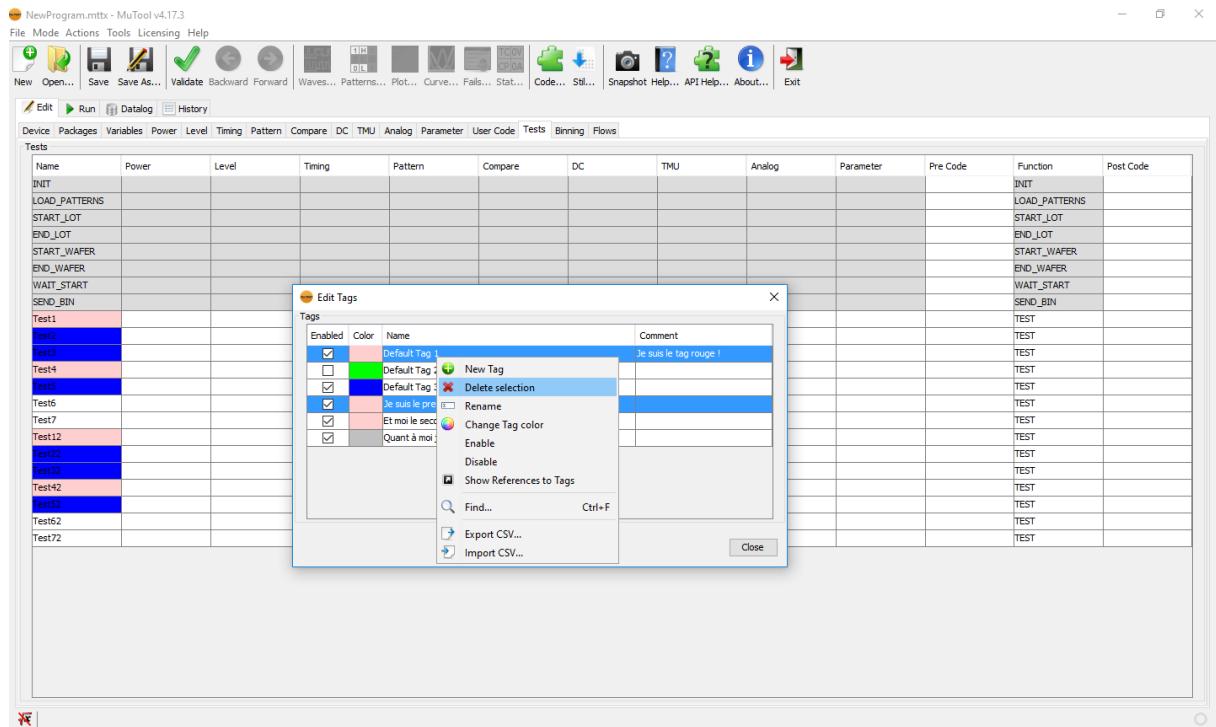


On choisit une nouvelle couleur.

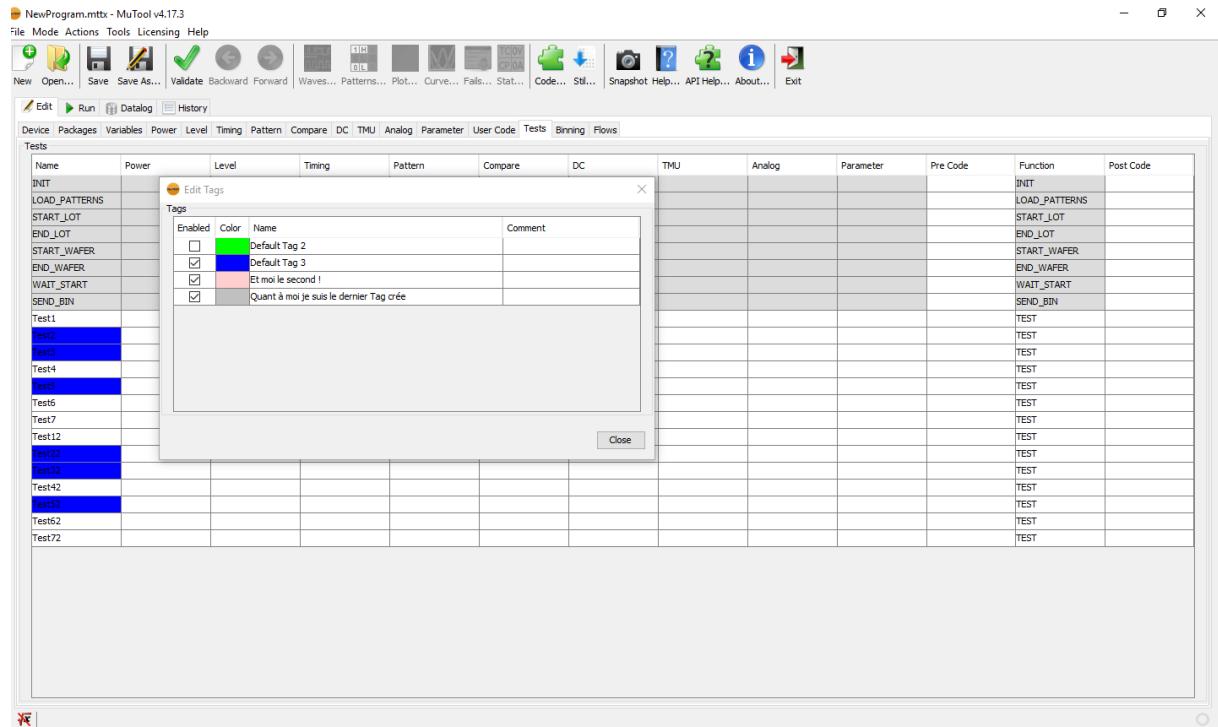


Notre couleur choisie est affectée aux Tags sélectionnées, et le Test Panel est rappelé afin d'actualiser les couleurs.

## Suppression :



Un Popup nous avertit les Tests possédant ce champ et qui va être supprimé.



Le Tag et ses références ont été supprimés.

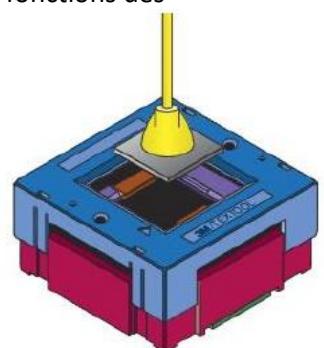
## VIII. Glossaire

---

1. **ATE** : Automated Test Equipment : équipement de test automatique, système permettant de soumettre un composant électronique à différents test pour définir si les caractéristiques sont correctes ou non.
2. **Salle Blanche** : Pièce où la concentration particulaire est maîtrisée afin de minimiser l'introduction, la génération, la rétention de particules au sein de la pièce



3. **DUT** : Device Under Test : dispositif à tester, ici les composants électroniques produits
4. **Handler** : maître : bras robot qui place automatiquement les composant dans un **socket<sub>5</sub>**.
5. **Socket** : cavité : conteneur où l'on dispose un composant, les sockets varient en fonctions des composants



6. **Prober** : sonde : utilisation de sondes qui vont se connecter directement sur le wafer.
7. **Gateway** : Portail : Désigne la passerelle réseau permettant de relier deux réseaux distincts.
8. **LoadBoard** : Plateau de chargement : outils qui permet de réaliser l'interface entre le composant et le testeur.
9. **Water cooling** : refroidissement du système via un système de circulation d'eau froide
10. **Pogo pin** : fine pointe en or permettant de réaliser un contact précis et efficace entre le LoadBoard et le testeur
11. **FPGA** : Field Programmable Gate Array : circuit Logique Programmable, cela permet d'appliquer des niveaux en sortie en fonctions des entrées dans le but de commandé les instruments du testeur.
12. **Pattern** : modèle : solution générique à un comportement. C'est en réalité une table des vérités qui permet d'appliquer un résultat en sortie en fonction des entrées.
13. **Release** : désigne la version d'un logiciel
14. **Software** : partie logiciel, programmation avec des langages de développement (C, Java ...)
15. **Hardware** : partie matériel, composante du testeur
16. **Firmware** : partie programmation bas niveau via les FPGAs
17. **Méthode Agile** : une méthode agile est une approche itérative et incrémentale ; les tâches vont s'effectuer petit à petit, par ordre de priorité, avec des phases de contrôle et d'échange avec le client.
18. **Mélée quotidienne** : court rassemblement équipe par équipe hebdomadaire afin de réguler les tâches du sprint en cours
19. **Régression**: désigne des fonctionnalités qui étaient antérieurement fonctionnelles ne le sont désormais plus.
20. **Update**: mise à jour du système, ajouts de nouvelles fonctionnalités, correctifs pour augmenter la stabilité
21. **RC**: Release Candidate : version Candidate, une version expérimentale (interne) de celle qui sera livrée en fin de sprint
22. **SC**: Sanity Check : test de santé : on vérifie les fonctionnalités les plus importantes du système
23. **Log**: « registre » : un Log contient des informations écrites par un logiciel. Lors de l'exécution d'une séquence test, il est possible d'activer l'enregistrement des données dans un fichier .txt/.log (fichier texte). Le Log contient alors des détails sur l'exécution des différents tests, ce qui permet de comprendre l'échec.
24. **Java**: langage de programmation orienté objet, connu pour sa portabilité sur différents systèmes d'exploitation (IOS / Linux / Windows) ainsi que sa documentation très fournit.
25. **IDE** : Integrated Development Environment : environnement de Développement : ensemble d'outils pour augmenter la productivité d'un développeur (ex : Visual Studio, Eclipse, NetBeans)

26. **Design** : Modification du visuel avec des outils graphiques de l'IDE

27. **Versioning** : versionnage : désigne le mécanisme qui consiste à conserver la version d'une entité logicielle quelconque. Cela permet de modifier simultanément un programme par plusieurs personnes et de bénéficier des nouvelles modifications si elles sont partagées.

28. **Class** : en programmation désigne un objet par des propriétés

29. **Héritage** : fait d'hériter des propriétés d'une autre Class.

30. **Getter** : méthode permettant l'accès des variables internes à une Class

31. **Setter** : méthode permettant la modification des variables internes à une Class

32. **Surcharge** : fait d'utiliser la même méthode mais avec des paramètre différents.

33. **Abstract** : méthode désignant l'utilisation d'une Class

34. **CSV** : format utilisé pour sauvegarder des données sous d'un tableau.

35. **XML** : fichier texte utilisé pour la configuration

36. **Override** : surmonter : en Java permet de réécrire une méthode d'une Class

	A	B	C	D	E	F	G	H
1	name	Virtual Memory	Working Set	Private Memory	Non-paged memory			
2	atieclxx	66068480	6533120	2588672	9736			
3	atiesrxx	33804288	4505600	1765376	7632			
4	audiogd	0356224	15941632	15011840	9424			
5	CCC	674254848	6746112	61571072	76332			
6	conhost	85331968	7323648	2846720	7920			
7	csrss	57634816	5296128	3284992	13568			
8	csrss	110956544	14876672	8941568	18986			
9	DCPSysMg	109682688	12632064	8765440	13208			
10	DCPSysMg	59002880	7610368	2768896	11408			
11	Dell.Conts	605573120	49803264	43466752	42724			
12	dupupdchk	116756480	10899456	4263936	15816			
13	dwm	201625600	50298880	38662144	23224			
14	explorer	45148832	113147904	88936448	80720			

```
["name","Virtual Memory","Working Set","Private Memory","Non-paged memory"],["atieclxx",66068480,"6533120","2588672","9736"],["atiesrxx",33804288,"4505600","1765376","7632"],["audiogd",0356224,"15941632","15011840","9424"],["CCC",674254848,"6746112","61571072","76332"],["conhost",85331968,"7323648","2846720","7920"],["csrss",57634816,"5296128","3284992","13568"],["csrss",110956544,"14876672","8941568","18986"],["DCPSysMg",109682688,"12632064","8765440","13208"],["DCPSysMg",59002880,"7610368","2768896","11408"],["Dell.Conts",605573120,"49803264","43466752","42724"],["dupupdchk",116756480,"10899456","4263936","15816"],["dwm",201625600,"50298880","38662144","23224"],["explorer",45148832,"113147904","88936448","80720"]]
```

Exemple d'un fichier CSV lu par Excel