
11 Avril 2017
9ème édition
JOURNÉE FRANÇAISE
DES TESTS LOGICIELS
LE TEST LOGICIEL, MAÎTRISER L'ÉTAT
DE L'ART DU TEST AVEC LE CFTL !
ISTQB
International Software
Testing Qualifications Board

RETOUR D'EXPÉRIENCE DU TEST DU MOTEUR DE COTATION DE LA BOURSE PAN –EUROPÉENNE EURONEXT ET DE SON SI EN MODEL-BASED TESTING

EURONEXT

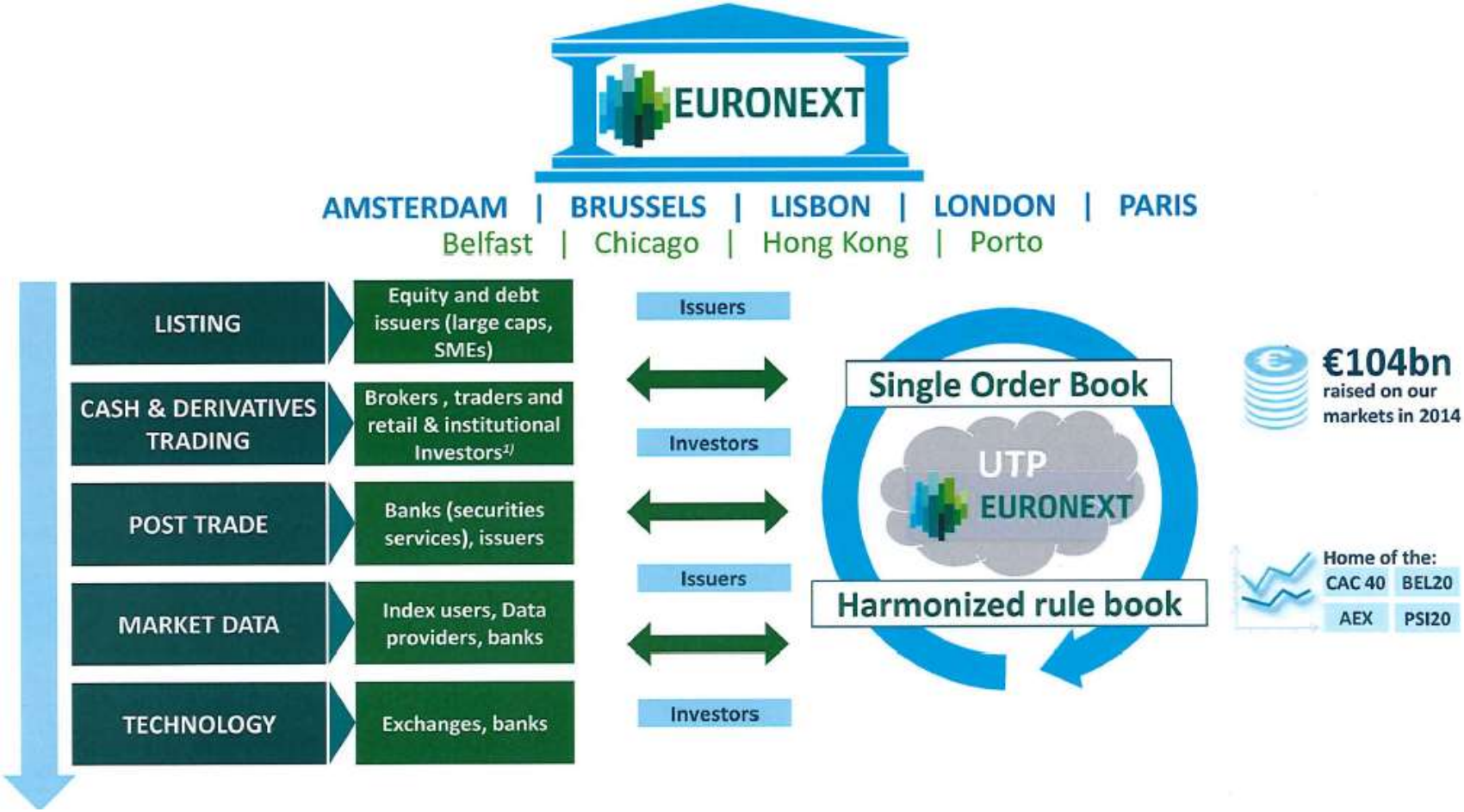
Thierry Schartle– Practices QA Leader
27 ans d'ancienneté chez Euronext (20 années d'expérience dans la Qualité)

Sandra Machon – IT Project Manager
10 ans d'ancienneté chez Euronext (12 années d 'expérience dans la Qualité)

Euronext QA Group :

- **2 Sites Paris and Porto**
- **2-3 personnes par équipes Agile**
- **50 Personnes**

EURONEXT – THE ONLY PAN- EUROPEAN EXCHANGE FOR THE REAL ECONOMY



…in multiple countries… …on a single order book… …and a common technology

EURONEXT TECHNOLOGY – KEY FACTS

113µs average customer roundtrip time when trading on UTP (on regulated cash markets)

2.2 billion messages per day

99,96% availability on our Equities platform in 2013 and **100%** on Derivatives

Internal matching engine roundtrip as low as **23µs**

526 member sites and 700 member lines connected to our systems

A leading-edge technology

1µs = one millionth (10−6 or 1/1,000,000) of a second

Contexte avant la mise du Model Based Testing

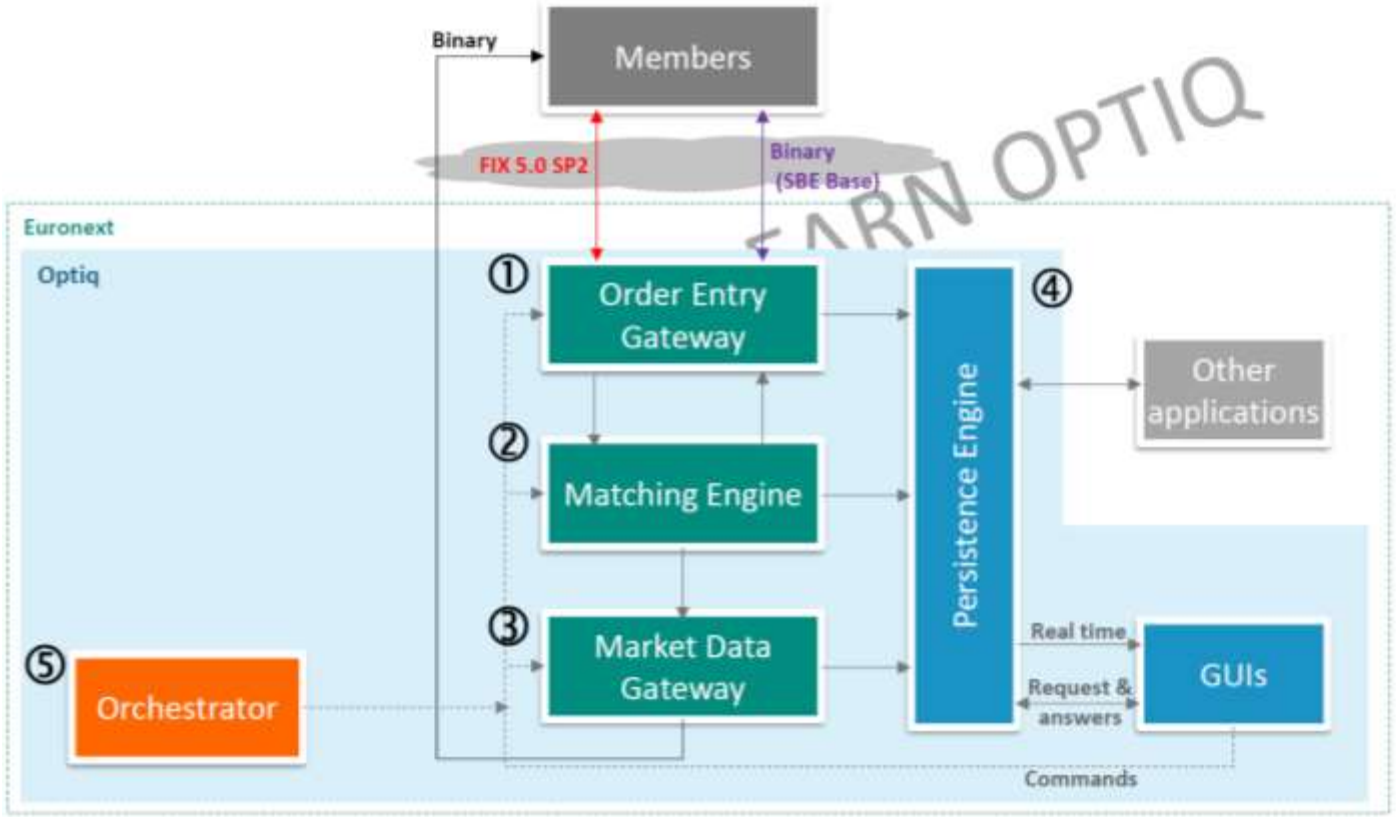
Phases principales du Cycle QA	Améliorations récentes	Faiblesses	Améliorations nécessaires
Planification, suivi et contrôle	Métriques sur la traçabilité entre les exigences et les cas de tests	La couverture n'est pas correctement mesurée, surtout en ce qui concerne la qualité de notre couverture.	Amélioration du suivi du notre couverture pour pouvoir améliorer la qualité
Analyse et conception des tests	La compréhension des exigences et des objectifs de test, a été améliorée par la mise en place d'Agile. Les cas de tests sont revus.	L'écriture des cas de tests est complètement manuelle et la maintenance en cas de changement d'exigence ou en phase de maintenance projet est très couteuse et diminue la qualité du référentiel de test. La couverture est limitée par le nombre de cas de tests (et de pas de tests). Il est difficile d'évaluer la qualité de la couverture.	Automatisation de la conception des cas de tests pour maximiser la couverture et gagner en flexibilité quant à la stratégie de la couverture
Implémentation et exécution des tests	Le taux d'automatisation est amélioré en continue.	La première exécution/automatisation est manuelle.	Construire un process entièrement automatisé
Evaluation des critères de sortie et clôture	Traçabilité	Traçabilité	Traçabilité

Comment le Model Based Testing peut répondre aux besoins Business ?

Améliorations nécessaires [Quality + Time To Market]	Model Based Testing Approach	Objectif couvert
Automatisation de la conception des cas de tests pour maximiser la couverture et gagner en flexibilité quant à la stratégie de la couverture	En analysant les user Stories et les specifications, les testeurs élaborent des modèles qui génèrent automatiquement les suites de tests associées à une stratégie de tests	☺
Amélioration du suivi du notre couverture pour pouvoir améliorer la qualité	La traçabilité entre les exigences et les tests est définie au niveau du modèle, les cas de tests générés sont alors liés aux exigences. La qualité des tests est améliorée car les modèles sont revus, ce qui est plus efficace que de revoir x000 cas de tests, et surtout l'objectif de revue est devenue atteignable et fait maintenant partie du process (QA, dev, BA). Le choix de la stratégie de tests permet de prendre en compte la maturité du produit et les risques.	☺
Construire un process entièrement automatisé	Basé sur les modèles, les cas de tests sont générés automatiquement en fonction de la stratégie de tests, avec les scripts d'exécution associés aux résultats attendus, cela permet d'avoir une comparaison automatique des résultats observés et attendus dès la première exécution.	☺
Traçabilité	L'industrialisation de l'ensemble des phases permet de créer un process QA transparent.	☺

OPTIQ TARGET ARCHITECTURE

Optiq is composed of 5 core components: ①the Order Entry Gateway (OEG), ②the Matching Engine (ME), ③the Market Data Gateway (MDG), ④the Persistence Engine/GUI and ⑤the Orchestrator



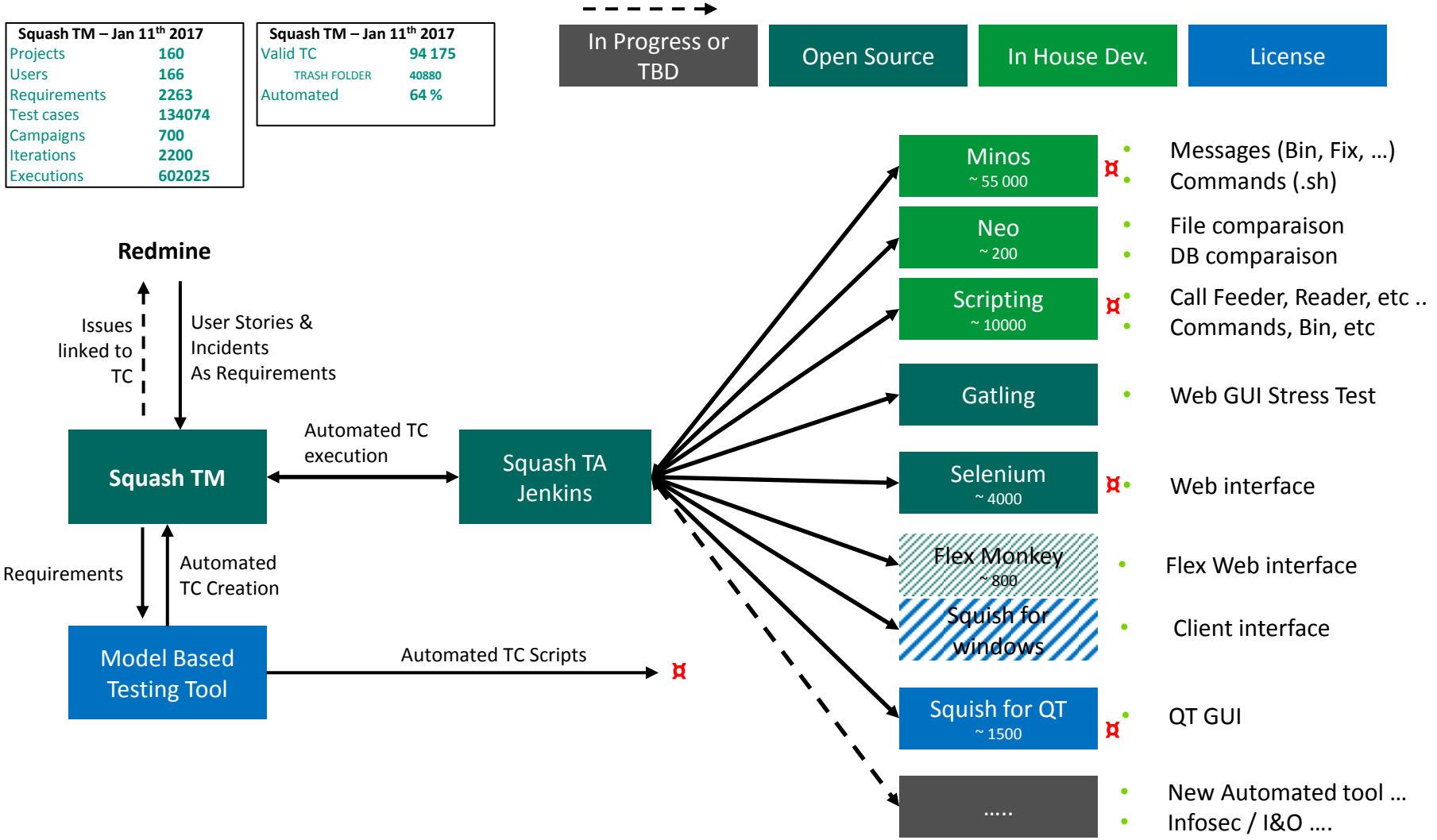
LE PROGRAMME OPTIQ

	DDM - Data Dictionnary Model	WS1 – OEG/ME Gateway messagerie privée Moteur de cotation	WS2- PE/GUI Moteur de persistence des data IHM	WS3 – MD Gateway messagerie publique
Enjeu de la QA	L'ensemble de notre système est basé sur l'échanges de messages, dont les structures sont définies dans le DDM. L'enjeu de la QA a été de mettre en œuvre une solution pour que tous les modèles « parlent » le même langage.	Couverture des algorithmes de trading, selon les différentes modalités d'ordres, les différents évènements, les règles fonctionnelles sous-jacentes.	Couverture sur l'affichage des propriétés selon les données d'entrées (flux ou fichiers), gestion des commandes, tests typiques d'interfaces (search, filtres, tree, formulaire, export…). Gestion de la persistence des données.	Vérification du format des messages, avec test des combinaisons, et règles de gestion pour les flux entre les système d'information des traders et le moteur de cotation, gestion du real time et du snapshot.
Organisation QA	QA Manager + 1 QA Analyst de chaque projet	QA Manager + QA Delivery Manager + QA Analysts	QA Manager + QA Delivery Manager + QA Analysts	QA Manager + QA Delivery Manager + QA Analysts
QA Tools	MBT Tool Python scripts	Squash MBT Tool Minos	Squash MBT Tool Squish Producer (Data injector) Consumer (Data consumer) Python scripts	Squash MBT Tool Feeder (Data injector) Reader (Data consumer) Python scripts

QA TOOLS

Squash TM – Jan 11th 2017	
Projects	160
Users	166
Requirements	2263
Test cases	134074
Campaigns	700
Iterations	2200
Executions	602025

Squash TM – Jan 11th 2017	
Valid TC	94 175
TRASH FOLDER	40880
Automated	64 %



NOTRE IMPLÉMENTATION MBT

LA MISE EN PLACE

- **Gestion de l'implémentation technique et des releases de l'outil MBT par notre équipe IT Tools**
- **Mise en place de deux Virtual Machines, chaque test analyst accède à l'une ou l'autre sur son propre workspace.**
- **Mise en place des formations avec l'éditeur MBT pour Paris et Belfast**
- **Accompagnement sur notre site par un expert outil MBT pendant 6 mois**
- **Lancement de l'utilisation de l'outil MBT sur le programme OPTiq (refonte de notre système, avec focus sur la chaîne de trading)**

NOTRE IMPLÉMENTATION MBT - 1

LE MODEL BASED TESTING POUR LE PROGRAMME OPTIQ

- **Quels modèles MBT en face de l'organisation du programme ?**
  - Choix : Un modèle pour une application
- **Et les tests d'intégration ?**
  - Choix : Intégration de chaque modèle MBT dans un modèle d'intégration
- **Comment gérer l'implication de plusieurs testeurs sur un même projet, sur un même modèle ?**
  - Choix : gestion des sujets isolés par chaque test analyst, puis merge des projets pour constituer un seul modèle qui portera l'ensemble du périmètre (avec gestion des versions sous GIT)

NOTRE IMPLÉMENTATION MBT - 2

ORGANISATION DU QA FRAMEWORK AVEC L'OUTIL MBT

- **L'organisation s'est faite par le biais de workshops hebdomadaires impliquant l'ensemble des utilisateurs. Les premiers risques identifiés portent sur notre capacité à modéliser et à maintenir la polyvalence de nos analystes de test (ie interventions de nos analystes de test sur l'ensemble du périmètre sous test).**
- **L'objectif principal est de synchroniser au maximum l'implémentation faite sur chaque projet par le biais :**
  - Des bonnes pratiques
  - D'une utilisation commune des structures et data
  - Du développement et du partage des compétences de modélisation
  - Du suivi par les QA Managers sur chaque projet

NOTRE IMPLÉMENTATION MBT - 3

DE L'ANALYSE DES EXIGENCES JUSQU'AU REPORTING DES CAMPAGNES DE TEST

- **Exigences de chaque projet saisies dans Redmine (User Stories) avec import automatique dans Squash, puis import des exigences de Squash vers l'outil MBT.**
- **La modélisation porte le lien entre les transitions et les exigences, puis les cas de tests générés et importés vers Squash permettent la traçabilité des exigences et le suivi de la couverture dans Squash.**
- **L'outil MBT génère la description des cas de tests (importés dans Squash sous forme de HTML) ainsi que les scripts de tests (pour les stimulations et les résultats attendus) selon l'outil d'exécution/automatisation utilisé.**

REFERENTIAL MBT

FROM XML TO DATA IN THE MBT TOOL

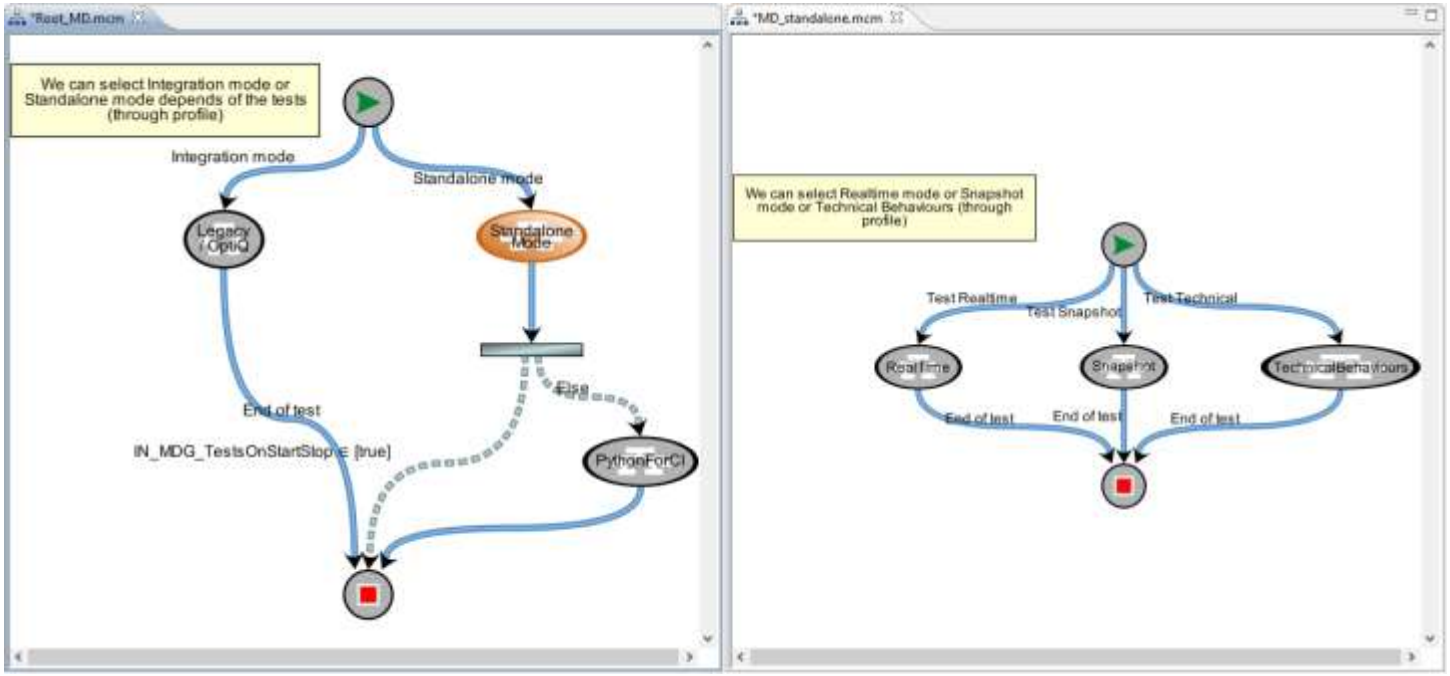
```
<enx:structure enx:numId="721" enx:name="MDBOrderUpdateData rep1">
        <enx:description>Repeating sub-structure of Order Update Event.</enx:description>
        <enx:structItem enx:name="FirmID">
                <enx:fieldLink enx:fieldID="Firm ID"/>
        </enx:structItem>
        <enx:structItem enx:name="SymbolIndex">
                <enx:fieldLink enx:fieldID="Symbol Index"/>
        </enx:structItem>
        <enx:structItem enx:name="ActionType">
                <enx:fieldLink enx:fieldID="Market Data Action Type"/>
        </enx:structItem>
        <enx:structItem enx:name="OrderPriorityTime">
                <enx:fieldLink enx:fieldID="Order Priority"/>
        </enx:structItem>
        <enx:structItem enx:name="OriginalOrderPriority">
                <enx:fieldLink enx:fieldID="Original Order Priority"/>
        </enx:structItem>
        <enx:structItem enx:name="OrderType">
                <enx:fieldLink enx:fieldID="Order Type"/>
        </enx:structItem>
        <enx:structItem enx:name="OrderPx">
                <enx:fieldLink enx:fieldID="Order Price"/>
        </enx:structItem>
        <enx:structItem enx:name="OrderSide">
                <enx:fieldLink enx:fieldID="Order Side"/>
        </enx:structItem>
        <enx:structItem enx:name="Quantity">
                <enx:fieldLink enx:fieldID="Quantity"/>
        </enx:structItem>
</enx:structure>

<enx:structure enx:numId="722" enx:name="MDBOrderUpdateData">
        <enx:description>MDPOrderUpdateData</enx:description>
        <enx:structItem enx:name="">
                <enx:structLink enx:structID="MDBHeader"/>
        </enx:structItem>
        <enx:structItem enx:name="EventTime">
                <enx:fieldLink enx:fieldID="Event Time"/>
        </enx:structItem>
        <enx:structItem enx:maxOccurs="255" enx:name="Orders" enx:occurenceCounter="Yes">
                <enx:structLink enx:structID="MDBOrderUpdateData rep1"/>
        </enx:structItem>
</enx:structure>
```

TYPE_INTERNAL_EVENTS_MDBOrderUpdateData	Structure	MDPO...	
name	TYPE_INTERN...	name	
MDBIdentifier	Numerical	MDBI...	
SnapshotIndicator	Numerical	Snaps...	
EMM	Euronext Mar...	EMM	
CentralOrderBook(COB)	Item		A 1
NAVTradingFacility	Item		A 2
PrimaryMarket	Item		A 3
Wholesales	Item		A 4
OffBook(TCS)	Item		A 5
OTC(TCS)	Item		A 6
OffBook(AtomX)	Item		A 7
SocieteGeneraleSystematicInternaliser(SI)	Item		A 50
NotApplicable(ForindexesandiNAV)	Item		A 254
EventId	Numerical	EventId	
BookOUTTime	Numerical	Book...	
MDBINTime	Numerical	MDBI...	
MDBOUTTime	Numerical	MDBO...	
EventTime	Numerical	Event...	
Orders	TYPE_INTERN...	Orders	
FirmID	Numerical	FirmID	
SymbolIndex	Numerical	Symb...	
ActionType	Identifies if th...	Actio...	
NewOrder	Item		A 1
DeletionOfIdentifiedOrder	Item		A 2
DeletionOfAllOrdersBySide	Item		A 3
ModificationOfExistingOrder	Item		A 4
RetransmissionOfAllOrders	Item		A 5
OrderPriorityTime	Numerical	Order...	
OriginalOrderPriority	Numerical	Origin...	
OrderType	Type of Order.	Order...	
Market	Item		A 1
Limit	Item		A 2
Pegged	Item		A 5
Markettolimit/Marketonopening	Item		A 6
OrderPx	Numerical	OrderPx	
OrderSide	Indicates the ...	Order...	
Buy	Item		A 1
Sell	Item		A 2
Cross	Item		A 3
Quantity	Numerical	Quan...	

MARKET DATA

- **2 modes de fonctionnement : Integration, Standalone. Zoom sur Standalone >> Real Time**



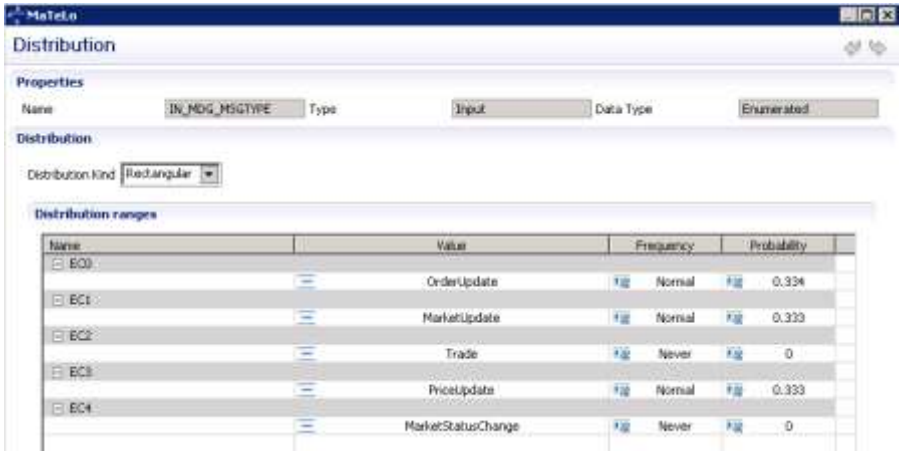
Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

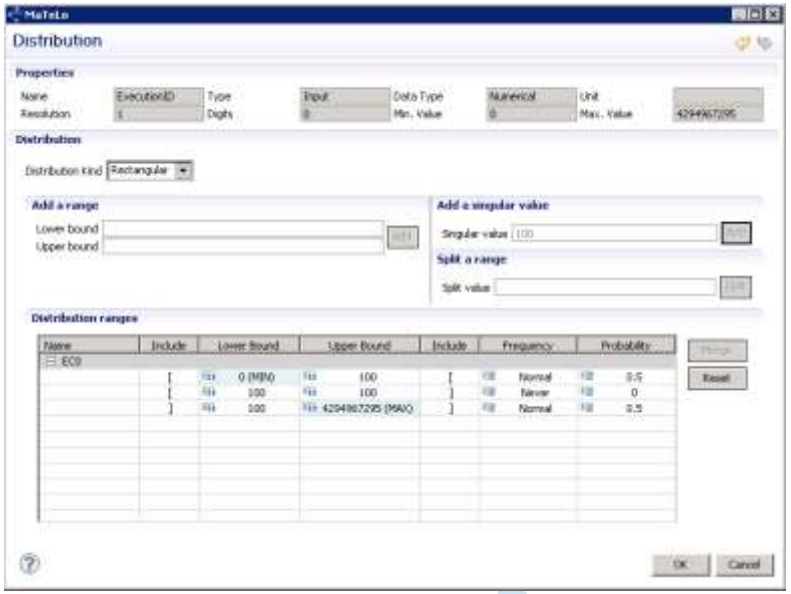
MARKET DATA

- **Zoom sur la chaîne « RealTime » gérant les messages valides en se basant sur les classes d'équivalence**

Validation des valeurs possibles basée sur les classes d'équivalence

Sélection du type de message en se basant sur les classes d'équivalence





Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

MARKET DATA

- **Zoom sur la chaîne « RealTime » gérant les messages valides**

Paramétrage des opérations de test associées sur les transitions

RepeatedGroup					
MDG_Python_MessageTyperep1-ForFeederForInternalPE-part2	MDG_Python_M...				
FirmID		Transition Input	0,1* IN_INTERNA...	Value	
ActionType		Transition Input	<E> IN_INTERNA...	Item Value	
OrderPriorityTime		Transition Input	0,1* IN_INTERNA...	Value	
OrderType		Transition Input	<E> IN_INTERNA...	Item Value	
OrderPrice		Transition Input	0,1* IN_INTERNA...	Value	
OrderSide		Transition Input	<E> IN_INTERNA...	Item Value	
Quantity		Transition Input	0,1* IN_INTERNA...	Value	

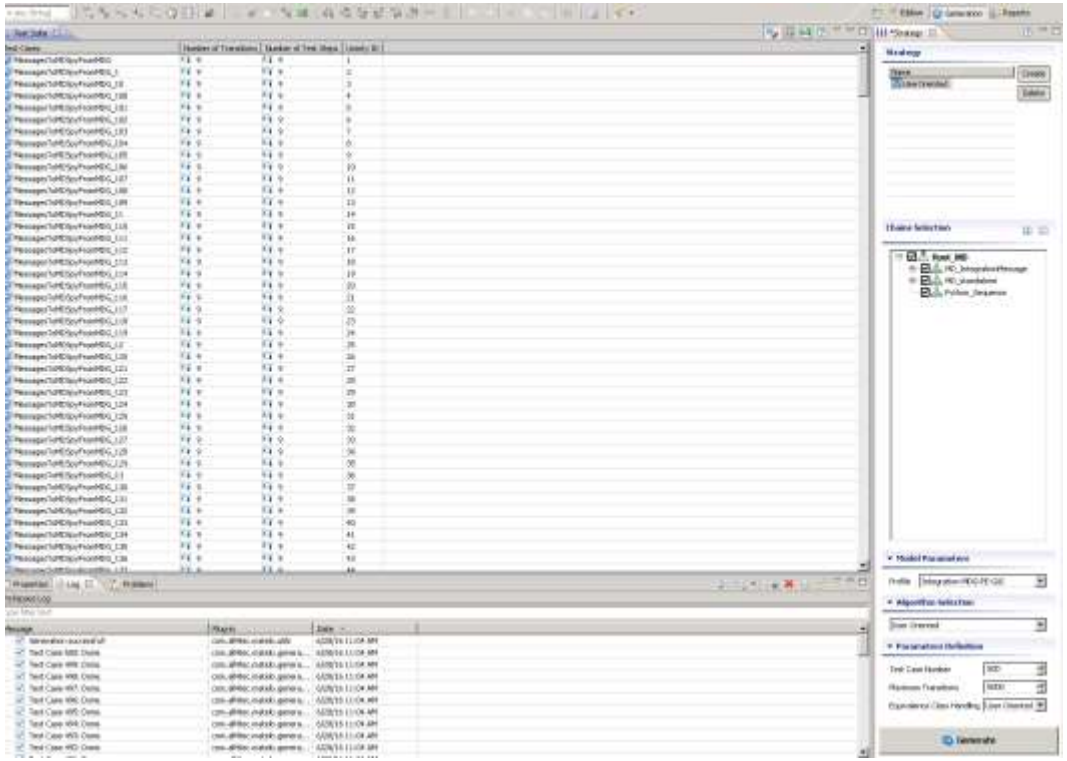
Ci-dessous le script décrit dans l'outil MBT

```
if(messageType == "OrderUpdate" and (tool == "Feeder" or tool == "PEIn")):
    part2 = "|FirmID=$FirmID,SymbolIndex=$SymbolIndex,ActionType=$ActionType,OrderPriorityTime=$OrderPriorityTime,OriginalOrderPriority=$OriginalOrderPriority,OrderType=$OrderType,Or
    fichier.write(part2)
elif(messageType == "MarketUpdate" and (tool == "Feeder" or tool == "PEIn")):
    part2 = "|UpdateType=$UpdateType,SymbolIndex=$SymbolIndex,Price=$OrderPrice,Quantity=$Quantity|"
    fichier.write(part2)
elif(messageType == "Trade" and (tool == "Feeder" or tool == "PEIn")):
    part2 = "|TradeTime=$TradeTime,ExecutionID=$ExecutionID,FirmID=$FirmID,MemberID2=$MemberID2,TradeType=$TradeType,TradeQualifier=$TradeQualifier,Price=$OrderPrice,Quantity=$Quanti
    fichier.write(part2)
elif(messageType == "PriceUpdate" and (tool == "Feeder" or tool == "PEIn")):
    part2 = "|PriceType=$PriceType,SymbolIndex=$SymbolIndex,Price=$OrderPrice,Quantity=$Quantity,ImbalanceQty=$ImbalanceQty,ImbalanceQtySide=$ImbalanceQtySide|"
    fichier.write(part2)
elif(messageType == "MarketStatusChange" and (tool == "Feeder" or tool == "PEIn")):
    part2 = "|ChangeType=$ChangeType,SymbolIndex=$SymbolIndex,EventTime=$EventTime,PhaseType=$PhaseType,StatusReason=$StatusReason,TradingMode=$TradingMode,OrderEntryQualifier=$Order
    fichier.write(part2)
```

Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

MARKET DATA

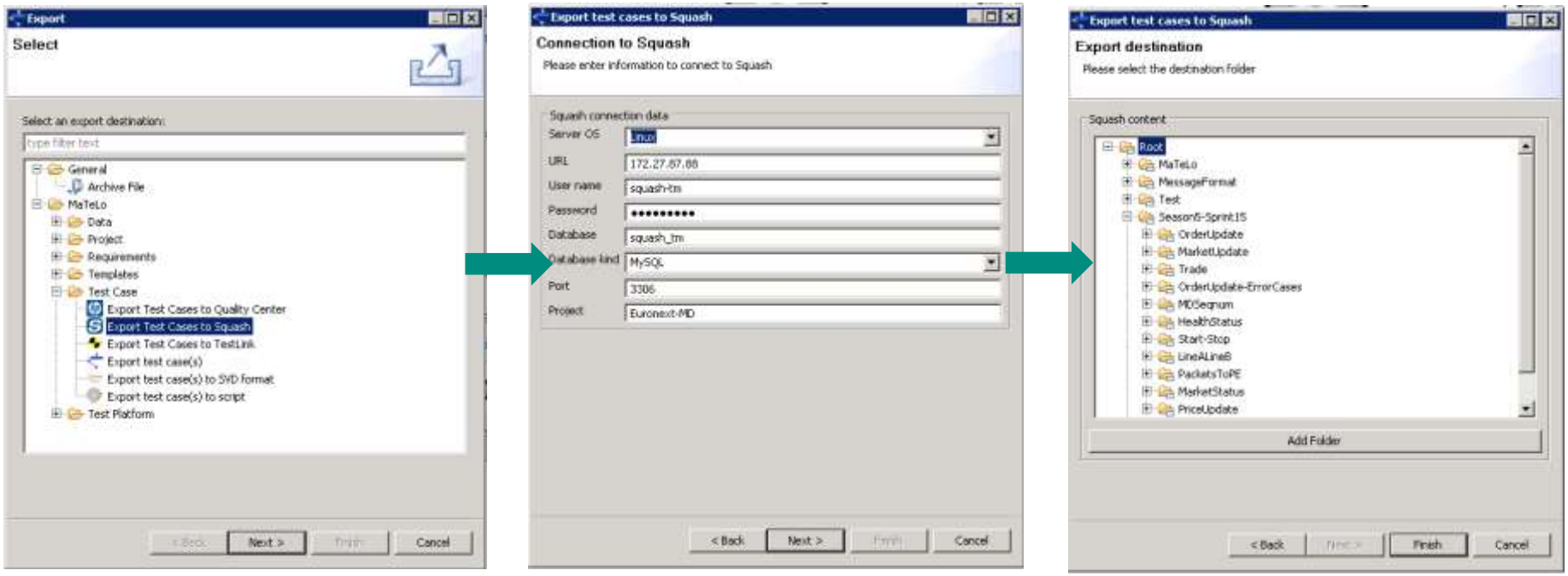
- **Génération de cas de test**



Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

MARKET DATA

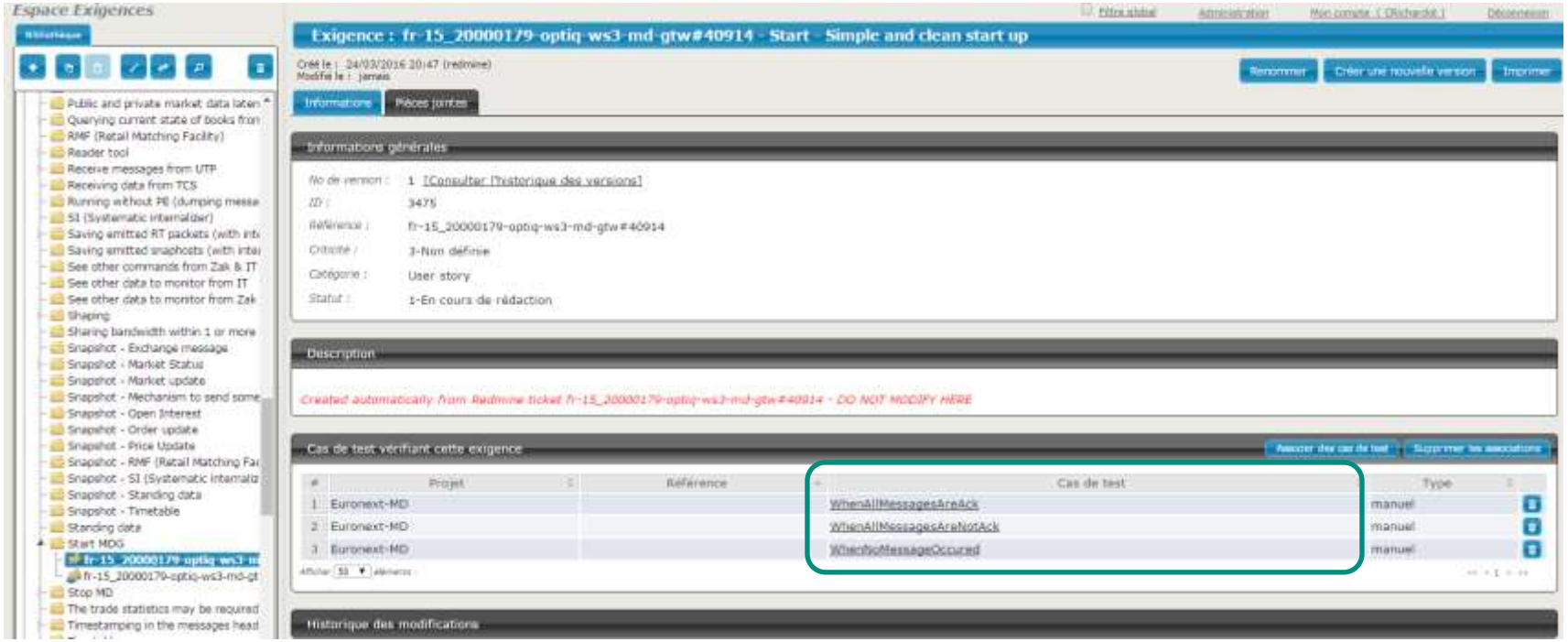
- **Export des cas de test vers Squash afin de faire le lien entre les cas de test générés et les exigences**



Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

MARKET DATA

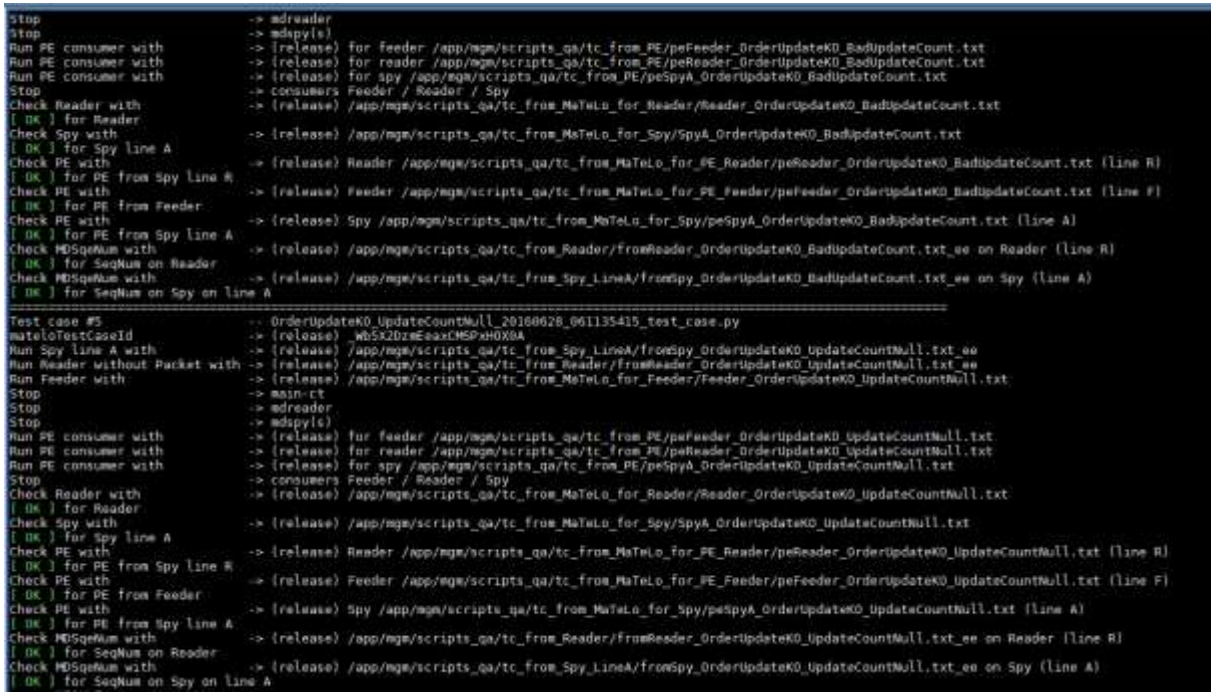
- **Visualisation des exigences liées aux cas de test**



Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

MARKET DATA

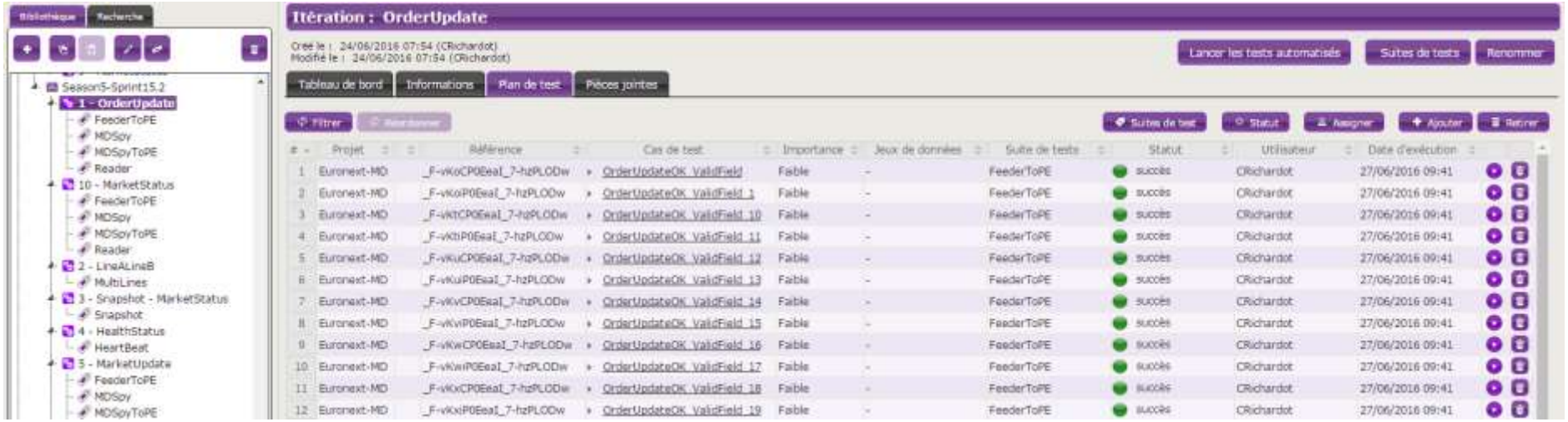
- **Pour chaque cas de test de l'outil MBT exporté au format « script », un fichier « python » est généré et exécuté sur l'environnement de QA.**



Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

MARKET DATA

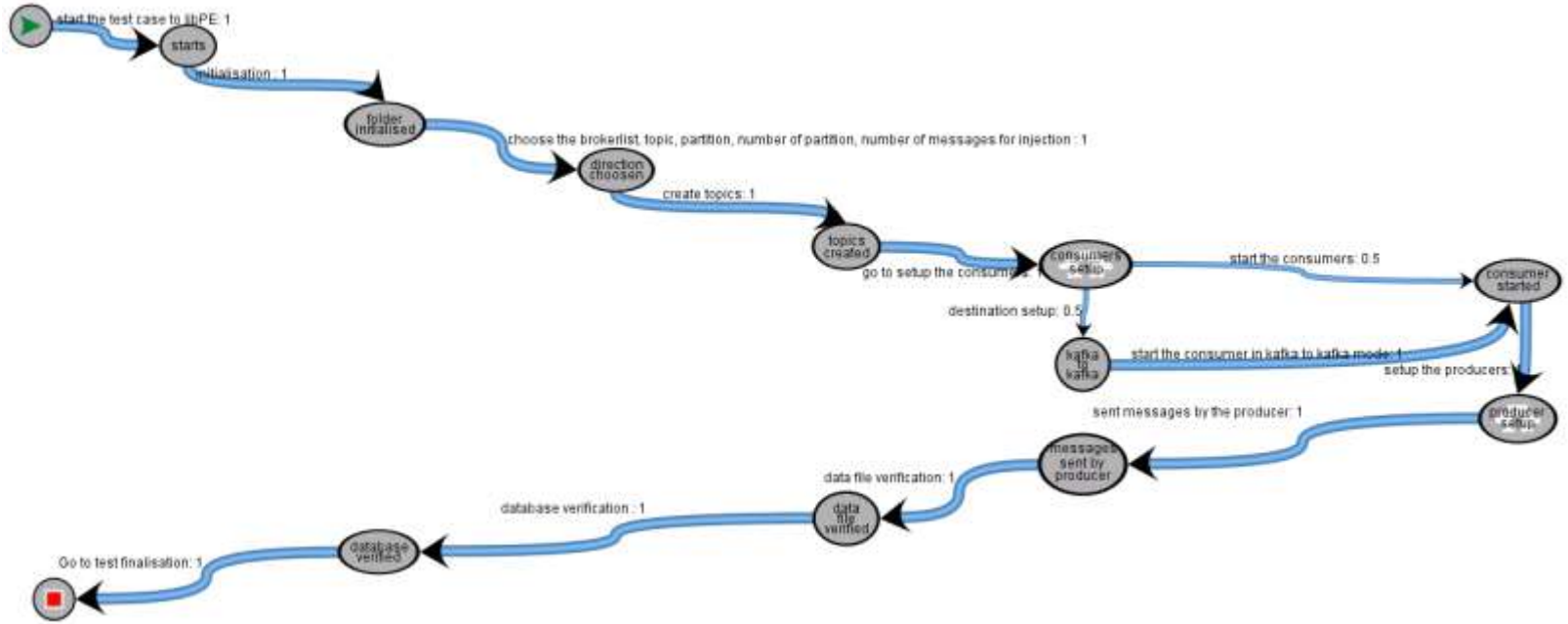
- **Exécution des scripts dans un environnement de test virtualisé**
- **Mise à jour des résultats de l'exécution des test dans Squash**



Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

PERSISTENCE ENGINE

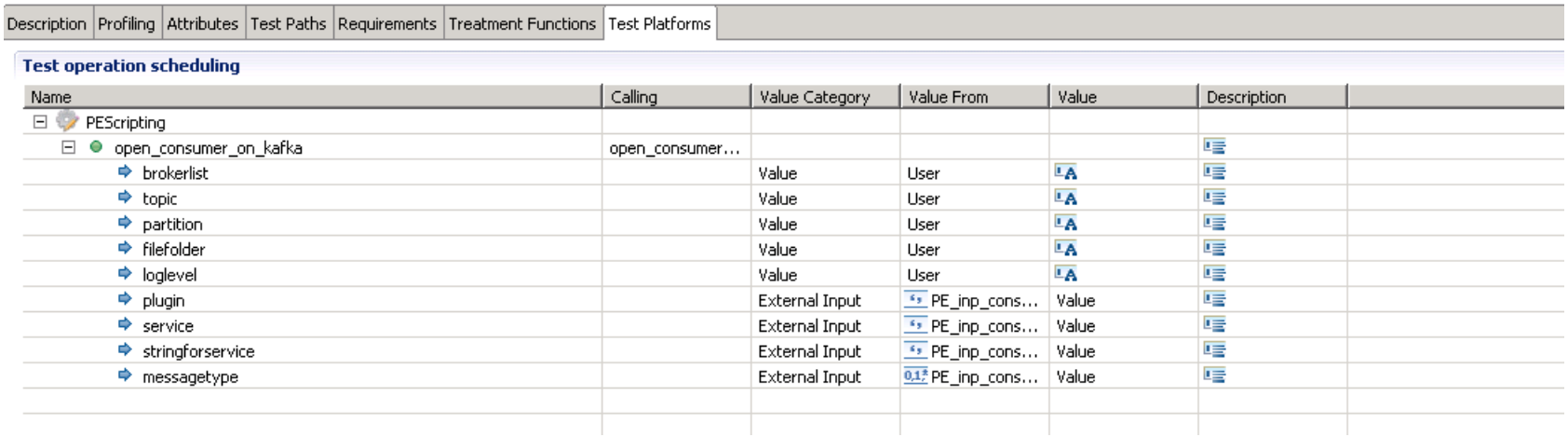
- **Objectifs : test de réplication, test de non corruption de données, test des plugins et services**



Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

PERSISTENCE ENGINE

- **Exemple de transition**



Name	Calling	Value Category	Value From	Value	Description
PEScripting					
open_consumer_on_kafka	open_consumer...				
brokerlist		Value	User		
topic		Value	User		
partition		Value	User		
filefolder		Value	User		
loglevel		Value	User		
plugin		External Input	PE_inp_cons...	Value	
service		External Input	PE_inp_cons...	Value	
stringforservice		External Input	PE_inp_cons...	Value	
messagetype		External Input	PE_inp_cons...	Value	

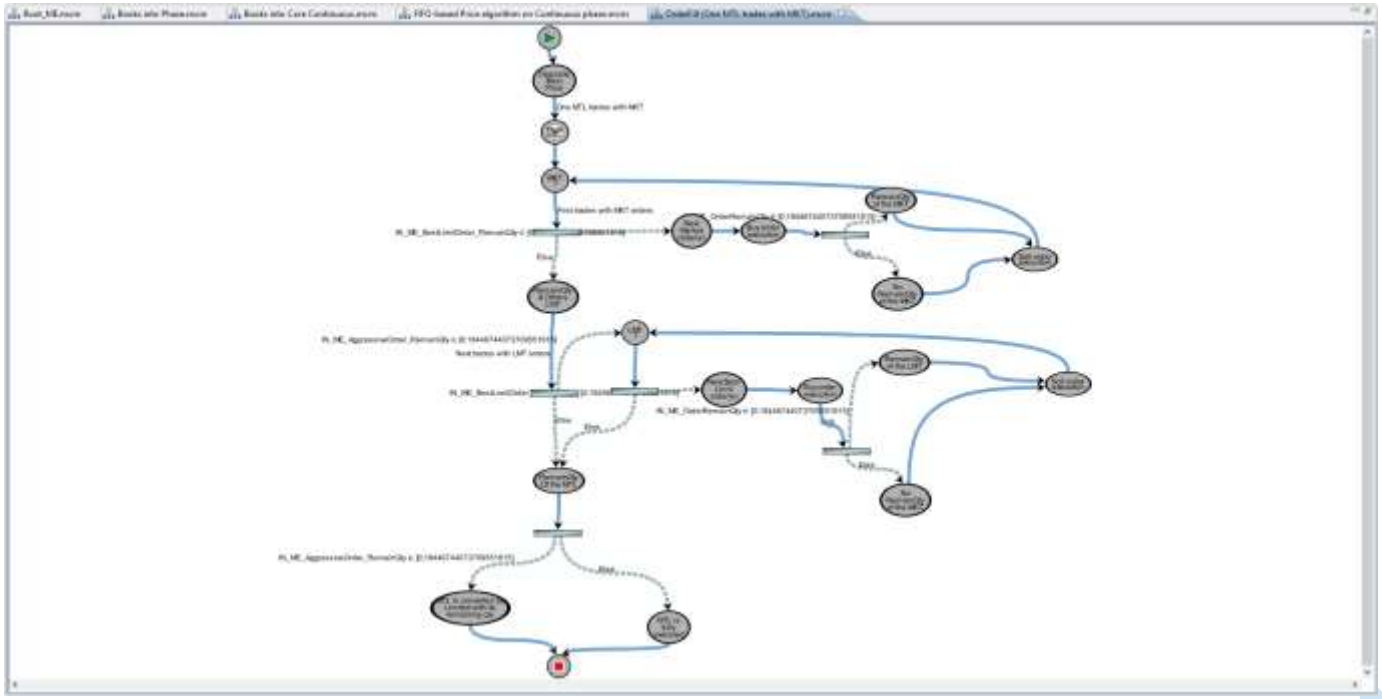
#open_consumer_on_kafka (brokerlist, topic, partition, filefolder, loglevel, mokorkafka, nbmessages, plugin, service, stringforservice, messagetype)

PE_python_script.open_consumer_on_kafka (brokerlist, topic, partition, filefolder, loglevel, "***$plugin", "$service", "$stringforservice", $messagetype)***

➔ A la fin, nous obtenons des scripts qui seraient **facilement compréhensibles fonctionnellement**

MATCHING ENGINE

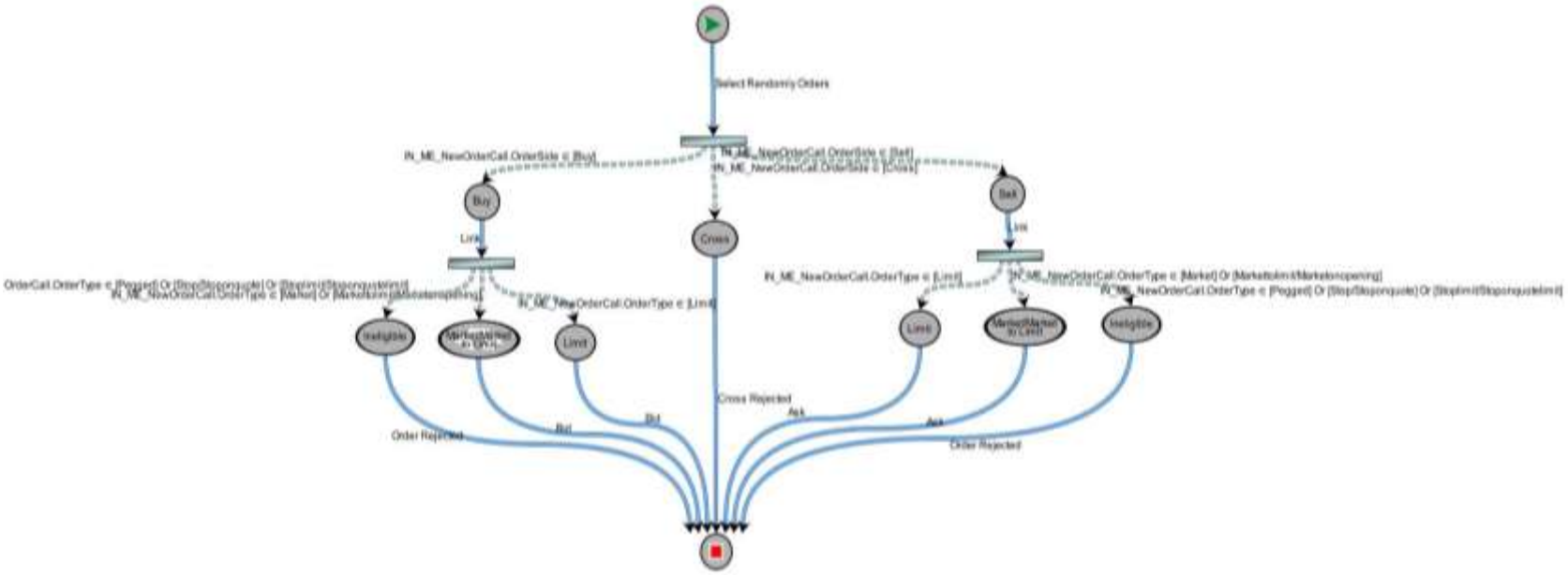
- **Execution d'un ordre « Market To Limit » dans un carnet ayant des ordres « Market » et/ou « Limited » >> modélisation déterministe par classe d'équivalences**



Modélisation | Génération | Export Squash | Génération script | Export et exécution Banc de Test

MATCHING ENGINE

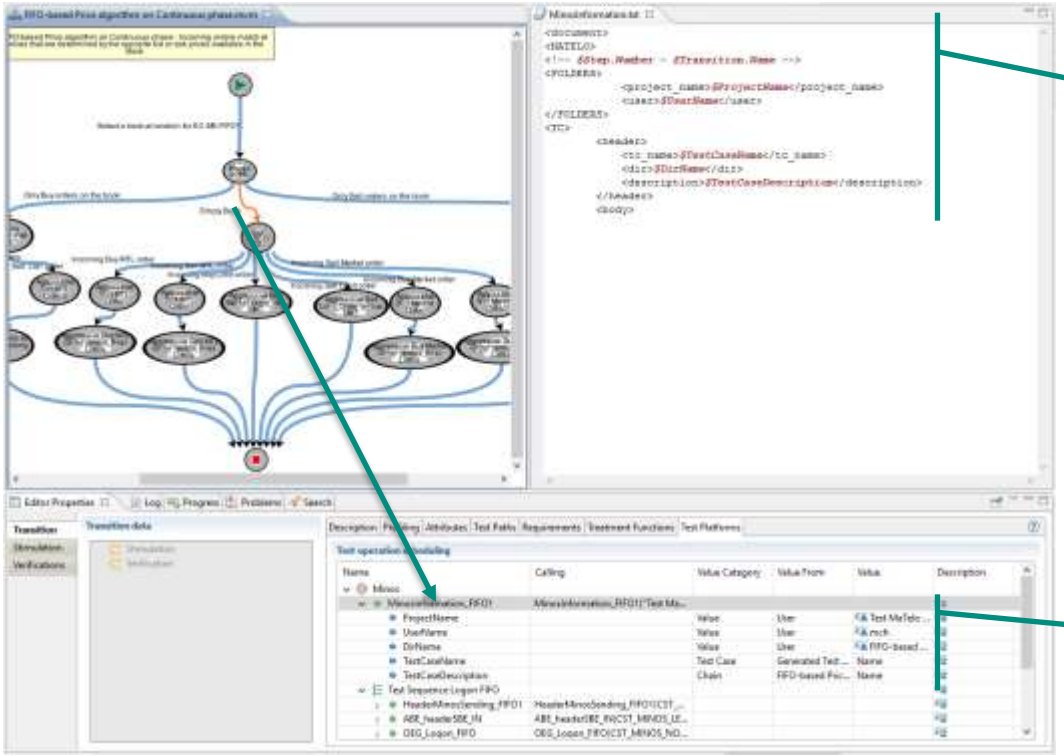
- **Exemple de modélisation d'un algorithme >> modélisation du comportement du système avec des stimulations aléatoires et la gestion des règles déterminant les résultats attendus**



Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

MATCHING ENGINE

- **Exemple d'utilisation d'une opération de test**

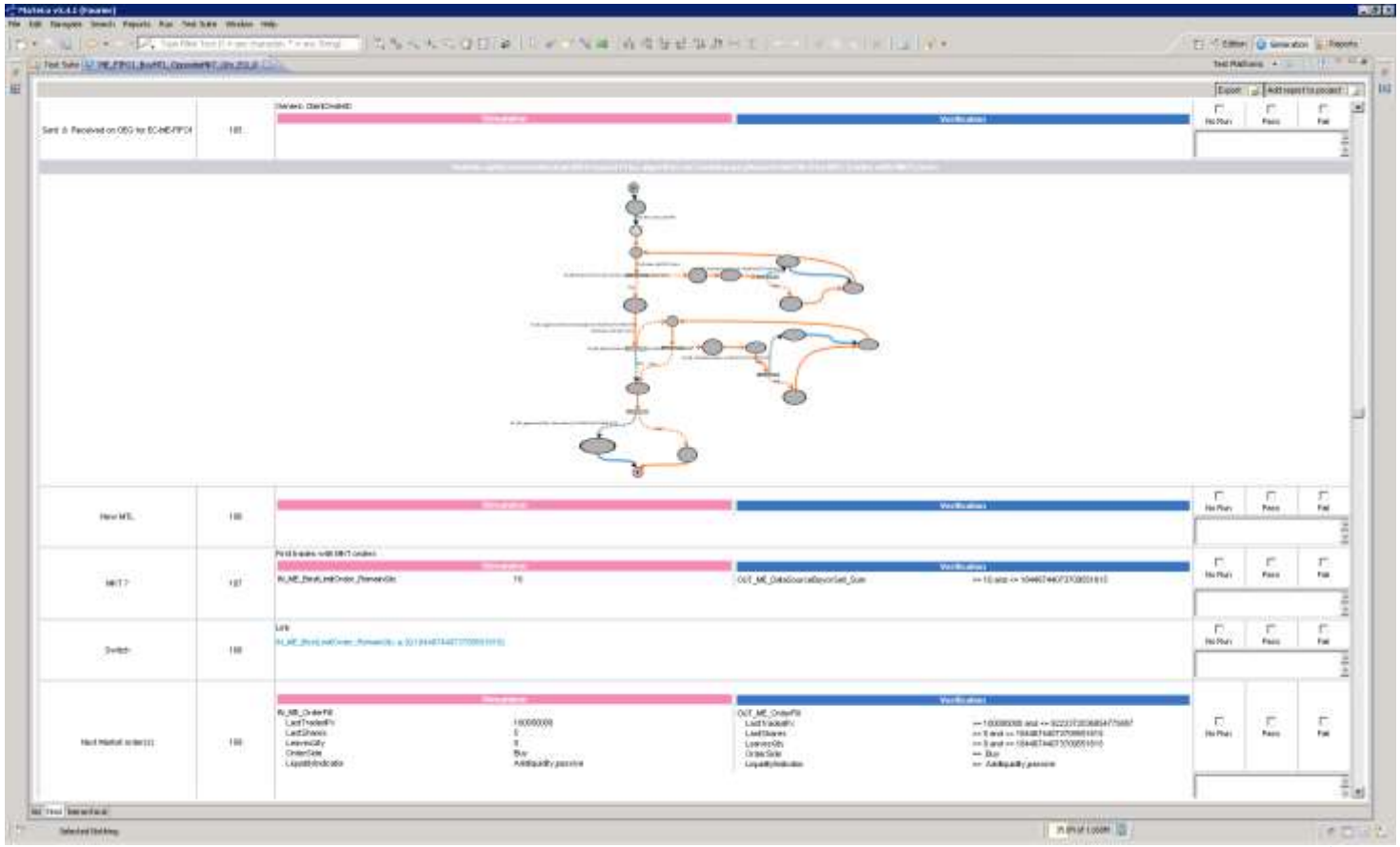


Fichier « script » correspondant à l'opération de test « MinosInformation _FIFO1 »

Association de l'opération de test avec remplissage des paramètres

Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

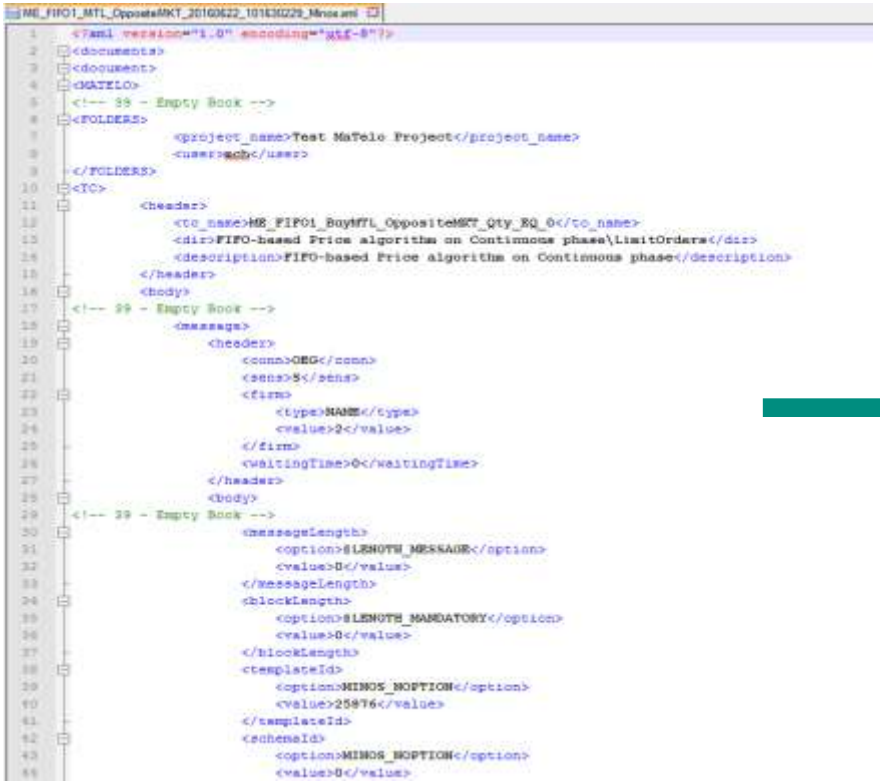
MATCHING ENGINE



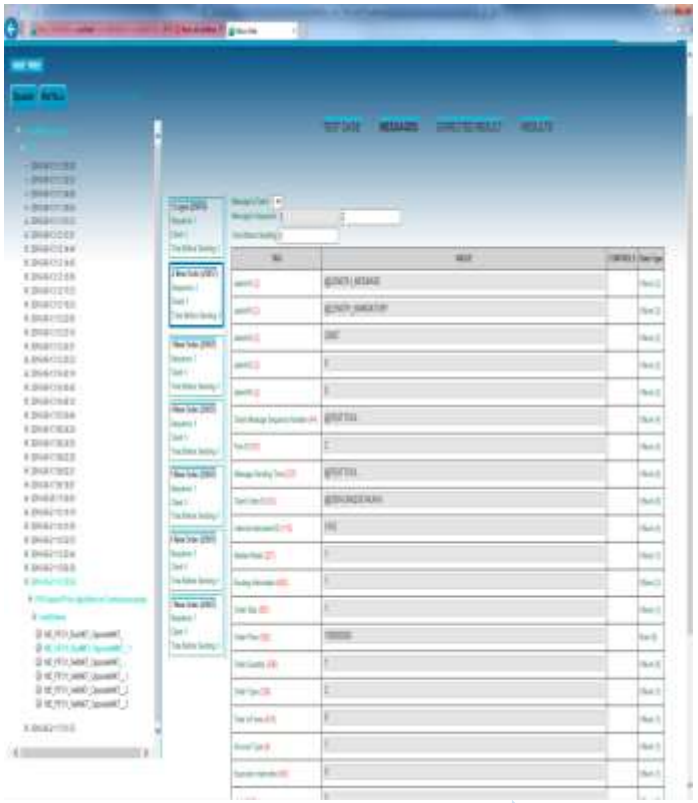
Modélisation | Génération | Export Squash | Génération script | Export et exécution Banc de Test

MATCHING ENGINE

Génération de script par l'outil MBT et import dans notre injecteur de test



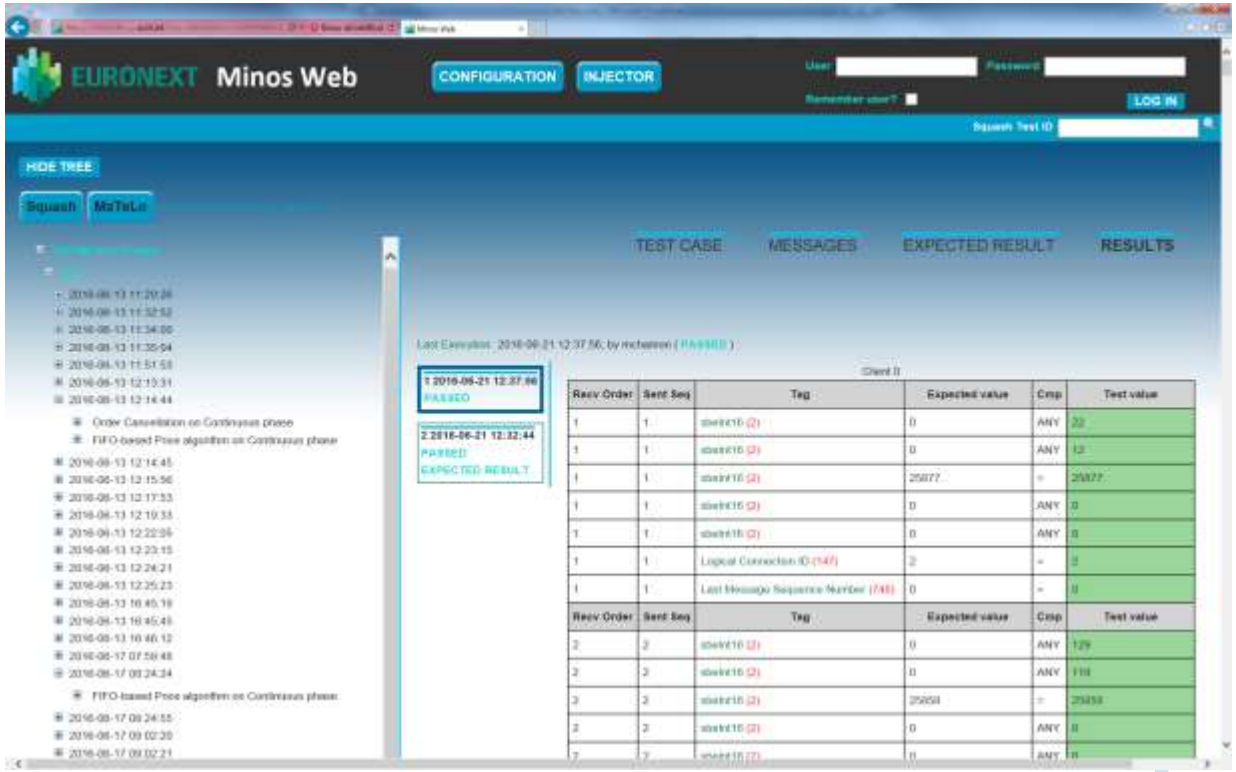
```
<?xml version="1.0" encoding="utf-8"?>
<documents>
<document>
<MATELO>
<!-- 39 - Empty Book -->
<FOLDERS>
        <project_name>Test MaTelo Project</project_name>
        <user>acb</user>
</FOLDERS>
<TC>
    <header>
        <tc_name>ME_FIFO1_BuyMTL_OppositeMKT_Qty_EQ_0</tc_name>
        <dir>FIFO-based Price algorithm on Continuous phase\LimitOrders</dir>
        <description>FIFO-based Price algorithm on Continuous phase</description>
    </header>
    <body>
<!-- 39 - Empty Book -->
        <message>
            <header>
                <conn>OEG</conn>
                <sess>5</sess>
                <firm>
                    <type>NAME</type>
                    <value>2</value>
                </firm>
                <waitingTime>0</waitingTime>
            </header>
            <body>
<!-- 39 - Empty Book -->
                <messageLength>
                    <option>$LENGTH_MESSAGE</option>
                    <value>0</value>
                </messageLength>
                <blockLength>
                    <option>$LENGTH_MANDATORY</option>
                    <value>0</value>
                </blockLength>
                <templateId>
                    <option>MINOS_NOPTION</option>
                    <value>25876</value>
                </templateId>
                <schemaId>
                    <option>MINOS_NOPTION</option>
                    <value>0</value>
```



Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

MATCHING ENGINE

Exécution dans Minos avec comparaison entre les résultats obtenus et les résultats attendus générés par l'outil MBT



Modélisation → Génération → Export Squash → Génération script → Export et exécution Banc de Test

CONCLUSION - SITUATION COURANTE

- 1 Projet concerné – refonte du SI – ~100 FTE
- 20 utilisateurs Test Analystes
- 4 Modèles majeurs en cours
- ~15000 TC Générés dont 95% automatisés
  - Commandes (Script Shell)
  - Python to Squish primitives – GUI
  - Export XML pour injecteur messages
- Partage des modèles pour 'validation'
  - Avec les Business Analyst
  - Intégration des TC dans le process de Continuous intégration

Next Steps

- Développeurs intéressés – Test Unitaires …
- Autres projets d'évolutions du SI 'legacy' …

CONCLUSION - ELEMENTS DE REUSSITES

- **Argumentaires Gestion du changement**
  - **R&D, innovation**
  - **Intégration dans un échosystem QA Tools riche et varié**

  => Value to QA

- **Processus de décision sur l'utilisation de l'outil MBT**
  - **Partage du PoC avec équipes Devs et PM**
  - **Validation tour de table Steering Comittee**

- **New Skills**
  - **IT ! Python, scripts, ….**

- **Gestion du risque**
  - **Training et accompagnement sur site**
  - **Réfèrent interne ultra motivé !**
  - **Suivi hebdo**

CONCLUSION – POINTS A AMÉLIORER

- **Fusion des Modèles**
  - **Difficultés pour fusionner les modèles a plus de 3 utilisateurs**
  - **Amélioration proposée: utilisation d'une data base**

- **Prise en main de l'Outil**
  - **Prévoir entre 1 et 2 mois d'utilisation pour une bonne prise en main**
  - **Bien choisir ces ressources**

ROI ATTENDU

Le ROI a été défini en prenant en compte l'implémentation d'Optiq, puis la réutilisation pour les nouveaux projets et BAU, Crs.

Optiq QA Estimations : ~6000 md	Pratiques avant Matelo	Optiq avec MaTeLo – 18 mois	Post Optiq – BAU & CRs
Analyse des Spec	10 %	10 %	10 %
Création des Tests	30 %		
Modélisation avec Matelo		30 %	10 % updates
Exécution des tests	30 %	30 %	20 %
Automatisation des Tests	20 %	10 %	10 %
Régression en Continue	10 %	10 %	10 %
Total	**100 %**	**90 %**	**60 %**
ROI attendu	**Rien pour les prochains 18 mois – implémentation d'Optiq**		**30 % to 40 % en 2018**