Le projet **DELIRE**

Développement par **E**quipe de **L**ivrables **I**nformatiques et **R**éalisation **E**ncadrée

NS7 – Rôle du Responsable des Tests

Du sacrifice à l'extase

« *Vous qui, comme moi, êtes Américains, ne vous demandez pas ce que votre pays peut faire pour vous, mais demandez-vous ce que vous pouvez faire pour votre pays. Vous qui, comme moi, êtes citoyens du monde, ne vous demandez pas ce que les États-Unis peuvent faire pour le monde, mais demandez-vous ce que vous pouvez faire pour le monde* ».
Discours inaugural de John Fitzgerald Kennedy, 20 janvier 1961

Quand je vous interroge sur votre rôle en cours, la réponse classique est « J'ai pour responsabilité de contrôler… » . Ce peut être le niveau de qualité, le respect du planning, l'exécution des tests
Comme il va y avoir des problèmes durant le projet (je rappelle qu'un projet est une suite d'emmerdes) vous risquez de souffrir, en voyant que vos objectifs n'ont pas été atteints. Et pourtant, le plan que vous aviez mis en place était béton, il suffisait de le suivre… Oui, mais ce ne fut pas le cas. Ô rage ! Ô désespoir !

Ce ne fut pas le cas parce que le nombre de plans à suivre était trop important, que l'esprit humain est assez peu multi-process et que, de toute façon, un projet c'est une suite d'emmerdes.

Il faut donc transformer le problème : plutôt que construire un beau plan puis vous demander comment le projet peut le faire pour vous, demandez-vous ce que vous pouvez faire pour le projet.

En d'autres termes, concevez votre travail comme une valeur ajoutée, soit pour le produit, soit pour le process. Et soyez heureux de voir que ce que vous avez proposé a permis d'améliorer la qualité du produit, ce qui permettra à long terme de proposer un produit industriel au client, et à plus court terme de construire une démonstration robuste pour la soutenance finale

Le rôle du Responsable des Tests
Mais qu'allait-il donc faire dans cette galère ?

Quand j'interroge les Responsables de Tests sur leur rôle, la réponse classique est « Je dois définir les tests, puis les exécuter »
Est-ce si sûr ?

Il y a 5 types de tests pour faire converger un produit :

- Les Unit Tests
  - Ils ont pour objectif de tester les composants de façon indépendante, les autres composants étant simulés, pour identifier et éradiquer les erreurs internes du composant de façon à ne pas polluer la suite des tests par des problèmes qui pouvait être identifiés très en amont.
  - Il est inutile de tester la résistance globale de l'avion si on n'a pas préalablement testé que chacune des pièces a été dimensionnée correctement et qu'elle tient la charge prévue par le calcul. Vous risquez sinon de faire s'effondrer l'avion par la responsabilité d'une seule pièce sous dimensionnée
  - Il en est de même pour les composants
  - Les Unit tests sont définis par chaque développeur responsable du composant
  - Comme ils devront être ré-exécutés à chaque correction d'erreur, il parait pertinent de les automatiser. I
- Les Integration tests
  - Ils ont pour objectifs de vérifier qu'un appel du composant A au composant B respecte l'interface proposé par ledit composant.
  - Ils sont réalisés en balayant le produit assemblé, de façon à passer par l'ensemble des interfaces définis
  - Cela ne sert à rien de tester que les portes permettent de faire rentrer 900 personnes dans l'avion en moins de15 minutes, si vous n'arrivez même pas à fixer ces portes sur la carlingue.
  - Ils ont en général définis par l'Architecte
- Les Function Tests
  - Ils ont pour objectifs de dérouler un scénario standard d'utilisation du produit
  - Durant ces tests, on va vérifier que les résultats proposés sont valides :
    - Arborescence de l'hôpital
    - Identité d'un personnel médical
    - Contenu d'un DMP
    - Respect du secret médical
    - Traçabilité …
  - C'est durant ces tests qu'on va vérifier que les performances, la convivialité, la documentation en ligne sont au rendez-vous.
  - Ils ont normalement dû être définis par le Designer durant la phase de Specification pour valider l'Architecture Fonctionnelle et l'Architecture Logique du produit
- Les stress tests
  - Ils ont pour objectif de valider que le produit se comporte de façon satisfaisante quand on n'utilise pas celui-ci de façon conforme

    - En gros, le produit ne doit pas planter, mais signaler à l'utilisateur l'erreur commise et potentiellement le moyen d'y remédier.
- Les tests d'acceptance
    - Ce sont les tests réalisés par le MOA lors de la réception du produit pour valider que celui-ci est conforme aux Spécifications et respecte les objectifs de qualité
    - C'est donc a priori une sous partie des tests fonctionnels

Comme vous le voyez, le Responsable des Tests n'a pas pour mission d'écrire ces tests ni de les exécuter, mais de construire un plan qui va maximiser l'efficience et l'efficacité des tests dans le processus de convergence du produit vers son niveau de qualité industrielle attendue.

Ce qu'o attend donc de son plan

- Des règles d'écriture des tests, en particulier des Unit Tests,
- Un plan d'exécution des tests, de collecte des informations, du nombre d'erreurs attendues, du coût de découverte et de correction, et une méthodologie pour valider la correction des erreurs.

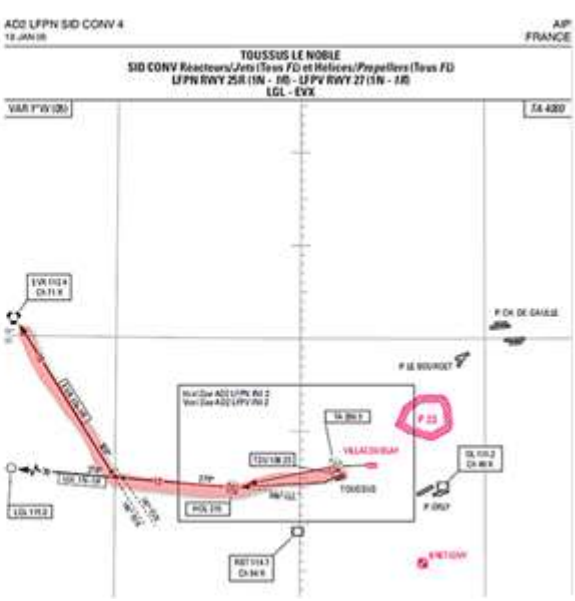
Dans les essais au sol puis en vol d'un nouvel avion, il y a une progression logique, qui demande un gros effort de préparation

Chaque essai en vol est minutieusement préparé

Plan de vol OCTAVE

```
PREPARATION ETOPS 180 MINUTES ** RR 3% ** ETP MEL-ETR-CTS

AFR   276/20.05.10   CDG-NRT    LFPG/RJAA    B777-200 / FGSPK
ATC: AFR276

CDB              HOR.DEP  11.15  BLOC DEP          DECOLL.
OPL              HOR.ARR  22.55  BLOC ARR          ATTERR.
OPL              TPS.HOR  11.40  TPS B/B           TPS.VOL

CARBURANT  PPV      REEL  TEMPS  DEOL  DAIR  VENT  FL
DEL.NRT    079800   ....  10.52  5430  5213  P020  320
DEG.MHD    002100   ....  00.17  0071        M010  080
R.RTE  3% 001800    ....  00.17
RES.FIN.   002900   ....  00.30                     PERFOS
CARBU SUP  000000   ....  00.00                     GE90-90B
TR.CARB.   000000   ....  00.00                     COM +1.00
ROULAGE    000800   ....  00.25                     CI120
TTL CARB   087400   ....  11.56
BILAN CARBURANT    -0204EUR              RESERVES ARRIVEE
CARBU DEFINITIF    ....                  AEROP. DEGAG. MHD /RJTT ...
E 1.355                                  DISTANCE DEG.    0071
                                         CARBURE. BLOC    .....  ....
M.B.CORR   150600  ....  LIMITATIONS     DELEST. DEG.     02100  ....
CHARGE     042000  ....                  CORRECTIONS      .....  ....
ZFW        192600  ....    195000        RES. FINS.       01900
CARBU TOW  086600  ....    135200        CARB. MINI.DEG.  .....  ....
TOW        279200  ....    280000        CARB. BLOC-DEG.  .....  ....
DELEST.    079800  ....                  CONSO.ARRIVEE    .....  ....
LAW        199400  ....    208400        CARB. ATTENTE    .....  ....
                                         PROC.SUP: NRT      00

ROUTE FMS : CDGNRTD      ROUTE OCATVE:RT OCD   OPTIMISATION : 2/2

LDG PISTE  2180 M
 CONSIGNES D'UTILISATION EN RONI
  RTE ETOPS . VIA FINLANDE / ARKHANGELSK
   CDG..NUBMO.UN874.VEKIN.UN879.SPY.UR12.JUIST.UN873.MAR.UT88.SOMOK.
UT82.NEBIN.UN746.KAKAT.R355.ULAM.A333.UMMM.R211.LANEP.B358.IOROD.
R347.OTC.R211.OOC..MILEY..SWAMP..KADMI..NRT

DEGAGEMENT AU DECOLLAGE : CDG   ....

DEGAGEMENT EN ROUTE DER 3% : **** / ****

****************************
*                          *
* VISA CDB                 *
****************************

SID SUR QFU : 08   CORRECTIONS
   08 / +00 NM / +00000 KG     26 / +15 NM / +00260 KG
   09 / -05 NM / -00090 KG     27 / -01 NM / -00020 KG
STAR SUR QFU : 16  CORRECTIONS
   16 / +00 NM / +00000 KG     34 / +00 NM / +00000 KG
AOUT-101 RC 5371 PLAN 0101 PAGE 02 OF 08 DATE 060720 TIME 100937
```

```
WPT       COORDONNEES    T.CUM  DEOL  MPLN-NEST/MSHRV DAIR  CONS-REST
                         DAT D.ISA        VENT      SR
  VIA     RVD RVA MAM    FL      D/T.SEG             VDOC      TROP

CDG       ............   00.00  3190 ...../...../.... 3492 000.9/000.2
  DCT     272 271 274    CLB       56/0.09

EVX       N49 01.9       00.09  3134 ...../...../.... 3438 004.3/076.8
          E001 13.3
  UT300   273 272 275    CLB       54/0.08

TOC       ............   00.17  3080 ...../...../.... 3383 007.5/073.6
                         -59 M05        358/037  2/2  ...../.....
  UT300   272 272 275    360        4/0.00            482       381

-LFRR.....BREST FIR

RUBIX   F N49 04.2       00.17  3076 ...../...../.... 3379 007.6/073.5
          W000 15.0      -59 M03        358/037  2/2  ...../.....
  UT300   272 271 275    360       36/0.05            482       381

SENLO     N49 05.0       00.22  3040 ...../...../.... 3342 008.4/072.7
          W001 10.7      -60 M03        356/033  2/2  ...../.....
  UM502   284 283 287    360       25/0.04            474       380

JUY       N49 13.3       00.26  3005 ...../...../.... 3306 009.1/072.8
          W002 02.8      -60 M04        350/023  2/2  ...../.....
  UM160   285 283 288    360       93/0.12            472       380

LIZAD   F N49 35.4       00.38  2912 ...../...../.... 3212 011.0/070.1
          W004 19.8      -61 M04        324/018  1/1  ...../.....
  UM160   303 303 308    360       13/0.02            466       380

MAKID     N49 42.9       00.40  2899 ...../...../.... 3198 011.3/069.8
          W004 37.4      -61 M05        325/018  0/1  ...../.....
  UM160   303 302 308    360       47/0.06            464       379

LND       N50 08.2       00.46  2852 ...../...../.... 3149 012.3/068.8
          W005 38.2      -62 M05        325/023  1/1  ...../.....
  UM160   301 300 306    360       31/0.04            456       376

INSAN     N50 22.7       00.50  2821 ...../...../.... 3117 013.0/068.1
          W006 19.4      -62 M06        328/034  3/3  ...../.....
  UM160   317 317 323    360       49/0.07            446       375

BOGMI     N50 59.9       00.57  2772 ...../...../.... 3063 014.1/067.0
          W007 12.3      -62 M06        325/038  4/5  ...../.....
  UM160   317 316 323    360       20/0.02            441       375

LEDGO   F N51 14.4       00.59  2752 ...../...../.... 3041 014.5/066.6
          W007 34.1      -63 M06        324/044  5/6  ...../.....
  UM160   316 316 323    360       27/0.04            436       374

ENTER     N51 33.7       01.03  2725 ...../...../.... 3012 015.1/066.0
          W008 03.6      -63 M06        317/048  6/7  ...../.....
  UM160   316 316 323    360       23/0.03            433       373

CRK       N51 50.4       01.06  2702 ...../...../.... 2986 015.7/065.4
          W008 29.7      -63 M07        315/049  6/7  ...../.....

AOUT-101 RC 0001 PLAN 0101 PAGE 03 OF 06 DATE 071212 TIME 140310
```



Il y a certes quelques heures en avion

Mais l’essentiel se passe en préparation et en dépouillement

.

Simplifions le problème dans DELIRE

Le travail
- d'écriture des Unit Tests,
- d'exécution de ceux –ci
- de correction des erreurs trouvées
- et de réexécution si besoin des Unit Tests

est incontournable

Le travail
- d'écriture des Function Tests,
- d'exécution de ceux –ci
- de correction des erreurs trouvées
- et de réexécution si besoin des Function Tests,

est incontournable

Par contre,
- A l'évidence vous ne ferez pas, par manque de temps, des Stress Tests
- Les Tests d'Acceptance, étant une sous partie des Function Tests, seront réputés exécutés et réussis, si ces derniers ont été exécutés et réussis
- Enfin, on peut se contenter d'exécuter une première fois les Function Tests sans s'intéresser aux résultats produits et d'appeler ces tests Integration Tests

Exprimé autrement, le document Plan de Tests doit se focaliser sur les Unit Tests et les Function Tests, en en définissant les règles d'écriture, et en en anticipant :
- Le coût d'exécution
- Le nombre d'erreurs attendus

Nombre de klocs

Il y a des standards de facto, construits sur l'expérience, du nombre d'erreurs

- 50 erreurs par kloc (Kilo Lines of Code)

ou du coût de préparation et de détection/correction pour un logiciel (supérieur à 50 klocs).

Etape opération	Préparation coûts	Dét./Répar. coûts
DR2	05H par Kloc	03H par bogue
I0	11H par Kloc	05H par bogue
I1	15H par Kloc	05H par bogue
I2	18H par Kloc	06H par bogue
U.T.	25H par Kloc	10H par bogue
Func.Test	40H par Kloc	24H par bogue
Comp.Test	60H par Kloc	24H par bogue
Syst.Test	20H par Kloc	35H par bogue

A défaut d'autres données, je vous propose de partir sur celles-ci pour faire un premier dimensionnement du coût des tests dans le projet DELIRE.

Vous l'aurez compris, il est nécessaire d'avoir une première idée du nombre de lignes de code prévues pour développer l'application.
Quand je demande aux étudiants un chiffe au pifomètre, leurs estimations s'étalent entre 1,000 lignes de code et 100,000 lignes de code. Gloups…

Je vous propose une approche un peu plus ciblée.

La productivité standard d'un développeur est de 10 lignes par jour !
C'est n'importe quoi, Antoine ???

Il faut bien comprendre ce que ce chiffre signifie.
10 lignes par jour, c'est durant tout le cycle de vie du produit : Spécification – Structuration – Réalisation – Convergence et… Maintenance.

On considère que dans un gros logiciel, la maintenance représente 60 à 80% de l'investissement : posons que le chiffre est 75%. Et nous ferons l'hypothèse que la maintenance n'augmente qu'à la marge la taille du code.
Donc on peut raisonnablement faire l'hypothèse que dans la partie développement du produit (Spécification – Structuration – Réalisation et Convergence), la productivité est de 40 lignes de codes.

Vous allez travailler 150 heures pour le projet environ. Soit 20 jours de travail à 7h30 par jour. Exprimé autrement, chacun devrait réaliser 800 lignes de code environ.

Vous êtes de 5 è 7 par groupe, on peut donc faire une estimation à la louche de 5000 lignes de code, 5 klocs.

Pour information, les quantités de lignes de code produites par les cohortes précédentes dans le projet DELIRE s'étalent grosso modo entre 2500 et 4000 lignes de code.
Pourquoi si peu ?

- Vous commencez à avoir une bonne expertise du développement logiciel, qui va être votre métier. Vous devriez donc plutôt faire mieux.
- Oui, si vous aviez été seul à coder, sur une technologie que vous maîtrisiez. Mais là vous cumulez les ennuis : domaine non maitrisé, développement à 6 ou 7, suivi 'd'une méthode de cycle en V.

Partons sur une hypothèse de 3000 lignes de code. 3 klocs

Des erreurs à trouver
3000 lignes de code, si c'est 50 erreurs par kloc, c'est facile : 150 erreurs à trouver.

Trop simple
50 erreurs par kloc, c'est durant tout le cycle de vie du produit : depuis la conception jusqu'à l'abandon dudit produit au profit d'un produit de nouvelle génération
Cela inclut donc :

- Les erreurs trouvées en phase de Spécification
- Les erreurs trouvées en phase de Structuration
- Et les erreurs à venir trouvées en phase de Maintenance, une fois le produit sur le marché.

Bref, le problème se pose de savoir quelle part des 50 erreurs par kloc on peut raisonnablement trouver ?

A priori, on est dans un exercice de divination. Ce qu'on appelle élégamment, en ingénierie, le pifomètre.

- Posons qu'un produit industriel a du éliminer 80% de ses erreurs. On passe à 120 erreurs à trouver
- Les coûts des phases de Spécification et de Structuration d'une part, de Réalisation et de Convergence d'autre part sont globalement équivalents en termes de travail. Posons qu'on doit trouver 50% des 80% d'erreur en phases de Réalisation et de Convergence. On passe donc à 60 erreurs à trouver
- Une erreur coûte grosso modo deux fois et ½ plus cher en coût détection/Réparation en Function Tests qu'en Unit Tests. Et la partie exécution des Integration Tests et des Function Test peut globalement être évaluée à la moitié du travail de la phase de Convergence. On peut donc penser qu'on trouvera globalement 50 erreurs en Unit Test, et 10 erreurs en Function Tests.

Maintenant se pose la question du cout de préparation, de détection et de correction des erreurs

Si vous prenez globalement les chiffres que je vous ai proposés :

- 50 erreurs en Unit Tests représentent 1250 heures de préparation et 500 heures de détection/correction
- 10 erreurs en Function Tests représentent 400 heures de préparation et 240 heures de détection/correction

Bref, au total : 2400 heures environ. Pour un projet de 900 heures au total c'est assez gênant.

Ce qu'on sait c'est que
Le cout de préparation des Function Test est déjà consommé, il s'agit de l'écriture des Function Tests
Le coût de préparation des Unit Test est composé du cout des STDs, déjà consommé, et du cout d'écriture des Unit Tests.

On va prendre le problème autrement, en faisant des hypothèses simplificatrices. Après avoir allumé un cierge à notre Dame du Bon Software pour supplier que le cout de préparation, détection et correction dans votre logiciel soit nettement plus faible qu'annoncé.
On va considérer que

- Le coût d'écriture des Unit Tests est équivalent au coût d'écriture du coding
- Le cout de correction des erreurs en Unit Tests est équivalent au cout d'écriture des Unit Tests
- Le travail majeur en phase de Function tests est engendré par la détection et la correction des erreurs.
- Ce travail représente 50 % du travail en phase de Convergence
- Et on peut considérer que le coût de détection représente 20% du coût, et la correction 80%

La phase de Réalisation représente 180 heures de travail

- 60 heures d'écriture des Unit tests
- 60 heures de correction soit 1h et 12 minutes par erreur

Le travail en phase de Function tests représente 50% de 180 heures soit 90 heures

- 18 heures d'exécutions des Function test
- 72 heures de correction des erreurs

J’ai rien compris
Normal, c’est une avalanche de chiffre

Je vous propose de refaire pas à pas le raisonnement de pifomètrage que je viens de faire, de vérifier que les chiffres proposés ont l’air raisonnable, puis de tout oublier, de faire votre modèle de pifomètrage personnel, et de le construire avec l’aide du Responsable de Planning et Budget d’une part, du Responsable Qualité d’autre part.

Mais Antoine, si on a tout faux ?
Je comprends que cela vous inquiète, et pourtant je considère que vous avez tort

D’une part, parce que l’objectif de DELIRE n’est pas d’avoir raison, mais d’apprendre ; l’important n’est ni le but ni le chemin, mais le cheminement. Ce qui est important c’est d’avoir posé les bases d’un modèle pour pouvoir avancer : à la fin du projet vous aurez emmagasiné suffisamment d’expérience pour mieux estimer les paramètres de votre modèle

D’autre part, parce que vous allez mettre en place une roue de Deming : rapidement vous aurez une bonne idée de la productivité de votre groupe en termes de découverte et de correction d’erreurs tant en Unit Tests qu’en Function, Il sera alors possible de recaler le modèle.

Bon courage.