Bonjour,

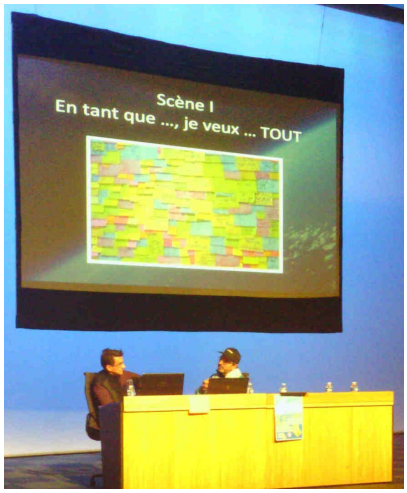
Cette année, Ausy a sponsorisé (tour.agiletoulouse.fr/sponsors) l'édition 2012 de l'Agile Tour de Toulouse (tour.agiletoulouse.fr), et cet événement associé à la souplesse de mon client qui m'a libéré pour l'occasion m'ont permis d'y participer moi aussi. Je remercie donc les 2 pour cette journée et je vais en partager avec vous les notes que j'ai pu en prendre.
***Disclamer :*** *Comme d'habitude, mes notes ne sont ni officielles, ni garanties (bien que je suis ouvert à toute remarque tant sur le fond que sur la forme … sinon, comment progresser ?)*

La journée a donc commencé tôt, très tôt (trop ?). Bref, Toulouse est à 2h30 de route et le matin-même, ce n'est pas une sinécure … même si ne n'avais pas à conduire (merci Laurence).

Rentrons donc dans les choses intéressantes.

**Accueil** (*Claude Aubry / Sponsors*)
Nous avons été accueillis au sein du centre de congrès Diagora de Toulouse (www.diagora-congres.com) par Claude Aubry[1] (www.aubryconseil.com). Selon son annonce, nous étions donc 477 participants inscrits. En somme, une fréquentation en parfaite adéquation avec la taille du centre de congrès dans l'idée d'un accueil confortable.
Il a donc offert une tribune méritée aux sponsors de l'édition sans lesquels un tel événement serait difficile (je ne suis pas convaincu que le prix de l'entrée couvre à lui seul les frais engagés dans un événement d'une telle ampleur).
Je retiendrai particulièrement cette citation *initialement attribuée à Alexandre Boutin*, mais qu'il a lui-même réattribuée immédiatement à Platon : « On peut en savoir plus sur quelqu'un en 1h de jeu qu'en une année de conversation ».
Il est aussi annoncé que la conférence accueillera hors programme des sessions d'Open Talk[2]. Il faudra que je me prête à cet exercice lors d'une prochaine édition, cela semble pouvoir être très enrichissant. Mais voilà, je suis arrivé avec un programme en tête et j'ai fais de mon possible pour le suivre.

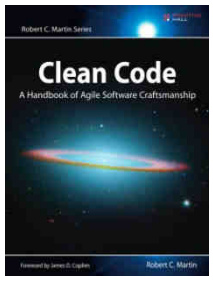
**Une question de valeurs** (*David Brocard / Thierry Cros*)
J'ai ensuite assisté à la session d'ouverture proposée par David Brocard (http://www.davidbrocard.org) et Thierry Cros (http://thierrycros.net) sous une forme … théâtrale. Un bijou de pédagogie.




- Le premier acte est une conversation entre un manager et un coach. Le premier présentant ses objectifs (meilleurs time to market, etc) et le second expliquant qu'il n'y a pourtant rien d'incompatible avec le respect des valeurs agiles en s'appuyant sur le manifeste (http://agilemanifesto.org/iso/fr/). Il y rappelle que toute pratique doit s'attacher à respecter ces valeurs, et fait un détour par l'analyse transactionnelle pour présenter brièvement les « niveaux logiques de Dilt-Bateson » (http://fr.wikipedia.org/wiki/Niveaux_logiques).
- Le second acte se déroule entre le PO (product owner) et le coach pour présenter quelques conseils de gestion des US (user stories). Le point le plus important étant qu'une US doit définir ses critères d'acceptation pour lever toute ambiguïté sur les attentes (éventuellement sous la forme « Etant donné ... Quand ... Alors ... »).
- Le troisième acte nous présente justement un exemple d'ambiguïté et de gâchis à travers une cérémonie de revue entre un développeur et le PO. Cette cérémonie a pu mettre en évidence le manque de communication et de critères d'acceptation qui ont amené à rejeter tout le travail réalisé sur la base de non-dits (*mais qui semblaient évidents au PO*).
- Le quatrième acte nous présente le principe de conception émergente à travers une confrontation entre deux développeurs. Le pari d'XP est qu'il coûtera moins cher de refactorer le temps venu que d'essayer de tout prévoir en Big Design Up front (http://en.wikipedia.org/wiki/Big_Design_Up_Front). Le tout est de s'appuyer sur un harnais de tests pour accepter le changement sans peur de régression. Sont aussi évoqués quelques principes plébiscités par XP tels que SOLID, KISS, le pair programming, ou encore la pratique de dojos, etc.
- Le cinquième acte est un premier retour entre un développeur et son manager sur la difficulté d'acceptation des changements induits par le passage à l'agilité et les contrebalançant avec les gains qui en ont été tirés. J'en retiens surtout une réplique du développeur : « Avant nous avions l'impression de graver du marbre, maintenant on a l'impression de façonner de la pâte à modeler. »
- Le sixième et dernier acte est une rétrospective entre le coach et le manager sur l'adoption de l'agilité et l'évolution induite du rôle de manager. Le coach en profite pour présenter brièvement la « théorie en X et Y » de McGregor (http://chohmann.free.fr/mcgregor.htm), et le manager souligne justement qu'il faut aussi composer avec des personnes qui souhaitent rester en X[3].

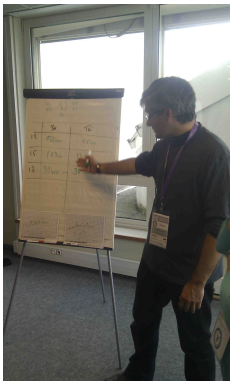
[1] Pour ceux qui ne le connaissent pas encore, il s'agit entre autres choses de l'auteur de « Scrum : Le guide pratique de la méthode agile la plus populaire ».
[2] Les sujets sont librement proposés via des post-its sur le stand puis sont traités par les participants par ordre de préférence
[3] Note personnelle : Mais faut-il vraiment les intégrer à des équipes agiles ? C'est un vrai risque !

**Clean Code en pratique** (*Jérôme Avoustin*)




J'ai enchainé par une présentation de pratiques de développement par Jérôme Avoustin (http://blog.avoustin.com/). Il s'agit là d'une introduction à l'œuvre « Clean Code » de Robert C. Martin[4] (http://www.amazon.fr/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882) bien dégrossie sur 1 heure de temps. C'est dense et bien des principes sont directement adressés au développement sous des langages objets modernes. Mais ce que j'apprécie dans ce type de présentation, ce sont les surprises cachées qui s'appliquent bel et bien à n'importe quel langage. Parmi eux, mon préféré qui ne semble pourtant pas être une évidence pour tout le monde : « Le code doit afficher clairement l'intention du développeur » car il faut penser à ceux qui repasseront sur ce que l'on produit. Jérôme va même plus loin sur ce principe en expliquant que si le code porte l'intention du développeur, les bugs auront beaucoup plus de difficultés à se cacher.
Bref, une présentation très riche dont voici les slides (http://fr.slideshare.net/jeromeavoustin/clean-code-en-pratique-14732075), et même si je ne suis pas personnellement en accord avec cette surenchère de l'abstraction qui mal maitrisée se paye beaucoup trop chère, je conseille vivement de prendre le temps de vous pencher sur les slides.

**Kanban par la pratique** (*Laurent Morisseau*)



Voici un atelier présenté par Laurent Morisseau[5] (http://www.morisseauconsulting.com/) qui nous présente principalement des outils de suivi et de prévision d'activité dans un contexte de flux tiré à travers une expérimentation d'un système Kanban et en s'appuyant sur la gestion des limites.
Il s'appuie pour cela sur un jeu commercialisé (http://getkanban.com/) dont je trouve personnellement le prix … *excessif*, mais qui permet de mettre en pratique l'utilisation d'un taskboard dans le cadre d'un système Kanban. Au-delà du fun, il s'agit d'une session riche en enseignements dont la cerise sur le gâteau réside en la proximité de Laurent qui nous fait partager ses retours d'expériences à travers les questions que le jeu peut lever.

**The testing strategy** (*Domenico Musto*)

Après une brève pause déjeuner (le retard de la keynote d'ouverture a décalé toute la matinée sans pour autant impacter l'heure des sessions de l'après-midi), je suis parti à la découverte d'un panorama des stratégies de test par Domenico Musto (http://att2012.agiletoulouse.fr/speakers/87). Dans l'immédiat, je n'ai pu trouver que les slides en italien (http://fr.slideshare.net/mimmozzo_/unit-tests-vs-end-to-end-tests), je vais donc reprendre ici les principales idées que j'ai retenu de cette présentation qui pour l'occasion était donnée en anglais.
La trame principale est donc une définition des différents types de tests.

- Les tests unitaires
  Ils permettent de tester des portions de code de façon isolée ainsi que l'interaction entre certains composants. Ils décrivent le comportement attendu du code, permettent de piloter le design et doivent impérativement être rapides à exécuter (sans quoi ils seront délaissés).
  Domenico conseille ici de ne tester que des interfaces publiques pour ne pas tomber dans le piège du test de l'implémentation plutôt que du comportement, et pour ne pas payer trop cher les refactoring. De plus, il déconseille d'abuser des mocks. Les 2 cas de figure identifiés pour leur utilisation sont :
  - la limite technique
  - la limite volontairement fonctionnelle

  Ce qui déclenche la rédaction d'un nouveau test unitaire est l'ajout d'une nouvelle fonctionnalité (ou identification de bug), et non la simple création d'une nouvelle classe.
- Les tests fonctionnels
  Il s'agit de valider l'interaction de l'ensemble des composants. Ils peuvent se permettre d'être lents mais doivent impérativement être indépendants les uns des autres (des context builder peuvent par exemple être utilisés pour tout ce qui relève du data fixture)
- Les tests de charge
  Il s'agit de tests de performances de l'application permettant d'identifier les limites sur des critères de volumes (de connexions, d'actions, …). Ils sont difficiles à écrire car impliquent la simulation de comportements en contexte de production.
- Les « soak tests »
  Je n'ai pas trouvé de traduction satisfaisante ☹. Quoi qu'il en soit, il s'agit de simuler l'utilisation du système sur de longue période pour identifier des problèmes (memory leak). Ils sont très longs à réaliser, et lancés manuellement pour une analyse approfondie.
- Les tests de bout en bout
  Il s'agit des tests d'acceptation. Ils doivent être compréhensibles d'un non-technicien mais sont difficiles à automatiser.
- Les tests d'intégration
  Il s'agit de tester des « facettes » de l'application pour valider leur intégration. Leur nature les rend nécessairement lents.
- Les tests exploratoires
  Il s'agit de tests manuels qui mettent à l'épreuve le système pour en découvrir des limites fonctionnelles.

[4] Plus connu sous le nom « Uncle Bob »
[5] Auteur de « Kanban pour l'IT - Une nouvelle méthode pour améliorer les processus de développement » (http://www.amazon.fr/Kanban-pour-lIT-am%C3%A9liorer-d%C3%A9veloppement/dp/2100578677)

Domenico insiste sur un point : pour que les tests, dans leur globalité, soient entretenus efficacement, un représentant du QA doit être intégré à l'équipe (*pratique préconisée de façon récurrente dans la sphère agile*).

Il donne ensuite une définition de la qualité pour un système. Il doit être :

- disponible
- modifiable
- performant
- sécurisé
- testable
- utilisable

Il présente ensuite un pyramide indiquant que :

- Toute application DOIT avoir des TU
- Toute application DEVRAIT avoir des tests d'intégration
- Peu d'applications possèdent des tests de bout en bout

Au cas où il n'aurait pas suffisamment martelé le message durant la présentation, Domenico conclut sur 2 points :

- Il ne devrait pas y avoir d'équipe de tests. C'est une cause d'échec. Les QA DOIVENT être intégrés dans l'équipe !
- Trouver un bon testeur est beaucoup plus difficile que de trouver un bon développeur

**Objectif Mars : qui pourra m'aider à peindre ma fusée** (*Hubert Gillon et son équipe*)

Voici un titre d'atelier surprenant présenté par Hubert Gillon (http://hubertgillon.wordpress.com) qui est assisté par 4 membres de son équipe. Il nous a présenté un atelier créé par Pierrick REVOL & Damien THOUVENIN (http://agileplayground.org/gameorama/objectif-mars-(fr).aspx) qui de prime abord semble compliqué … très compliqué. Cela justifie la présence d'un facilitateur par table de 5 joueurs. Ma table était facilitée par Cyril Aigoin (http://fr.linkedin.com/pub/cyril-aigoin/a/232/a39) qui a su nous faire intégrer les règles tout en nous laissant nous autogérer. Il faut effectivement quelques itérations pour rentrer dans le jeu, et à cet effet, la première est intégralement guidée par le facilitateur (et c'est nécessaire). Les choses intéressantes commencent donc à la seconde itération.

Derrière la complexité initiale du jeu, se révèle une richesse vraiment bluffante. Le nombre de principes de Scrum mis en évidence à travers une partie est juste inégalée vis-à-vis de l'ensemble des ateliers que j'ai pu côtoyer jusqu'ici (*au prix d'une facilitation intensive : merci encore Cyril*).

Je vais me contenter de lister quelques éléments du jeu que j'ai particulièrement appréciés :

- le déroulement de l'itération est pilotée par un tirage de dés. Classique me direz-vous. Mais l'utilisation de 3 types de dés différents (4 faces / 6 faces / 8 faces) pour représenter les différents niveaux de compétence des membres de l'équipe me semble d'une justesse rare. Ils mettent en avant que la productivité d'un débutant est nécessairement plus limitée que celle d'un expert, tout en gardant le reflet que l'expert peut aussi ne pas être constamment au top.
- l'existence dans le jeu de différents niveaux des membres de l'équipe, et surtout l'intégration de la notion de formation permettant d'intégrer dans les objectifs de production leur montée en compétence.
- la mise en évidence de l'impact sur la dette technique de l'exploitation de ressources insuffisamment formées ou compétentes.
- l'intégration de la notion de dette technique tout au long de l'atelier et la mise en évidence du coût prohibitif de sa mauvaise gestion.

Voici mon aperçu de l'Agile Tour de Toulouse. Je n'ai malheureusement pas pu assister à la keynote de clôture d'Alexandre Boutin (www.agilex.fr), mon taxi (*merci encore Laurence*) ayant des impératifs horaires. J'espère très vite trouver une rediffusion.

En attendant, j'ai nombre d'idées à suggérer/essayer en tête, et donc un seul conseil à prodiguer pour les prochaines conférences de ce type : allez-y, c'est enrichissant !