



## Une bonne dose d'agilité au cœur de votre équipe.

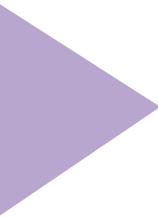
La recette Visual Studio 2012 pour des projets maîtrisés



Microsoft

[visualstudio.fr](http://visualstudio.fr)





**Une bonne dose d'agilité  
au coeur de votre équipe.  
La recette Visual Studio 2012  
pour des projets maîtrisés**

# Sommaire

---

<b>Avertissement .....</b>	<b>4</b>
<b>Introduction .....</b>	<b>5</b>
<b>L'évolution de l'ALM chez Microsoft .....</b>	<b>6</b>
<b>Le challenge des développements modernes .....</b>	<b>7</b>
1. Les différentes méthodes de travail .....	7
2. Les méthodes agiles .....	10
3. Le cas de Scrum .....	10
A. La perspective d'une équipe Scrum .....	11
B. La perspective du Product Owner .....	12
<b>Un sprint à travers les outils .....</b>	<b>13</b>
1. Choisir une méthode de travail .....	13
2. L'équipe.....	16
3. La gestion du backlog .....	17
4. Les storyboards : bien décrire une fonctionnalité .....	20
5. Planifier un sprint .....	23
6. Le nouveau flux de travail du développeur .....	27
7. Vous avez dit tests unitaires ?.....	35
8. Le besoin d'intégration continue .....	37
9. Test Manager 2012, un outil agile .....	40
10. Le besoin de feedback .....	44
A. Feedback produit.....	44
B. Feedback client.....	45
11. Suivre l'avancement d'un sprint .....	47
<b>Etre agile dans les nuages.....</b>	<b>51</b>
<b>Conclusion .....</b>	<b>52</b>
<b>A propos du Microsoft Technology Center .....</b>	<b>53</b>
<b>Annexe : Références techniques .....</b>	<b>54</b>

# Avertissement

---

Ce document s'adresse aux testeurs, aux architectes, aux concepteurs et développeurs, ainsi qu'aux chefs de projet qui souhaitent pouvoir mettre en œuvre une méthode agile avec les solutions Microsoft.

Ce document est fourni uniquement à titre indicatif. MICROSOFT N'APPORTE AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, À CE DOCUMENT. Les informations figurant dans ce document, notamment les URL et les références aux sites Internet, peuvent être modifiées sans préavis. Les risques d'utiliser ce document ou ses résultats sont entièrement à la charge de l'utilisateur. Sauf indication contraire, les sociétés, les entreprises, les produits, les noms de domaine, les adresses électroniques, les logos, les personnes, les lieux et les événements utilisés dans ce document sont fictifs. Toute ressemblance avec des entreprises, noms d'entreprise, produits, noms de domaine, adresses électroniques, logos, personnes ou événements réels serait purement fortuite et involontaire.

## Auteur

Vivien FABING et Etienne MARGRAFF sont tous les deux consultants formateurs dans la société de services Infinite Square, spécialisée dans les technologies Microsoft. Etienne est également reconnu Microsoft MVP (Most Valuable Professional) sur Visual Studio.

Stéphane GOUDEAU est Architecte au Microsoft Technology Center

Benoit Launay est chef de produit Visual Studio chez Microsoft France.

# Introduction

---

Jadis considérée comme risque majeur sur les cycles de développement associés à des méthodes traditionnelles, la notion de changement fait maintenant partie intégrante des processus de développement. Elle suppose la mise en œuvre, sur des cycles de développement courts, de méthodes dites « agiles » permettant un développement itératif et incrémental en favorisant la collaboration des équipes dans une recherche continue d'une solution répondant aux attentes du client.

Parmi ces méthodes, Scrum est sans doute la plus populaire. Couplée à l'utilisation des outils Visual Studio, elle devient redoutable de simplicité et d'efficacité.

L'objectif de ce document est de vous donner un aperçu de ces éléments.

## L'évolution de l'ALM chez Microsoft

Avant 2005, Microsoft proposait un ensemble d'outils qui mis bout à bout permettaient déjà de gérer un projet de développement. SourceSafe, Microsoft Excel, Microsoft Project, étaient autant d'outils permettant de planifier, suivre l'avancement et conserver de l'information. La problématique liée à ce mode de fonctionnement est que l'information est répartie dans plusieurs outils, sans liens les uns avec les autres. Il est difficile d'obtenir une vraie traçabilité de ce qui se passe et des différents échanges.

En 2005, Microsoft propose une solution intégrée permettant de répondre à l'ensemble des problématiques liées aux différentes phases de création d'un logiciel. Ceci se traduit par l'apparition d'une gamme de produit : Visual Studio Team System 2005 et plus particulièrement d'un outil qui la compose : Team Foundation Server 2005. L'objectif principal de cette version était d'unifier les aspects de gestion de projet et de gestion de code source tout en prenant en compte différents types de profils et les outils clients associés. La force principale de cette solution est de permettre aux différents intervenants du projet de continuer à utiliser les mêmes outils qu'auparavant : Excel pour la gestion des anomalies, et Project pour la planification par exemple. La différence ? Les données sont centralisées dans le serveur TFS.

La version 2008 a permis d'asseoir un certain nombre de fonctionnalités en prenant en compte l'ensemble des demandes d'améliorations.

La version 2010 a été une véritable évolution. Elle inclut un nombre important d'autres profils et d'autres aspects du processus de création logiciel. Désormais les testeurs sont réconciliés avec les développeurs. Il existe maintenant un outil complet de gestion de campagne de tests et de rapport d'anomalies. Le Lab management fait également son apparition et permet de simplifier grandement les aspects de provisionnement d'un environnement de validation d'une application en cours de développement.



Figure 1 : Fonctionnalités de la gamme de produit

Les problématiques de développement évoluent sans cesse. Tous les jours, de nouvelles approches sont imaginées, et de nouveaux besoins apparaissent. Les outils se doivent d'évoluer en conséquence et de permettre de prendre en compte ces nouveautés. C'est la direction que prend Microsoft en proposant très régulièrement de nouvelles versions et en l'enrichissant de pack de fonctionnalités et d'outils complémentaires tels que les Power Tools.

La version 2012 de la gamme de produits Visual Studio prend en compte les tendances actuelles. Les outils n'ont jamais été aussi simples et naturels à utiliser. L'agilité est une composante majeure de la nouvelle version de Team Foundation Server avec laquelle désormais un utilisateur n'a plus besoin de se poser la question : « Comment faire ce que je veux avec l'outil ? ».

# Le challenge des développements modernes

## 1. Les différentes méthodes de travail

Concevoir et réaliser des systèmes informatiques sont des tâches complexes, qui supposent la mise en cohérence de multiples ressources humaines, logicielles et matérielles souvent interdépendantes. Pour appréhender et formaliser cette démarche sur la totalité du cycle de vie de ces systèmes, sont apparues progressivement un certain nombre de méthodes.

Apparue en 1956, la première d'entre elles, le modèle en cascade (« Waterfall ») proposait un modèle en séquence avec les étapes suivantes : Conception, Initiation, Analyse, Design, Construction, Tests, Production, Implémentation et Maintenance.

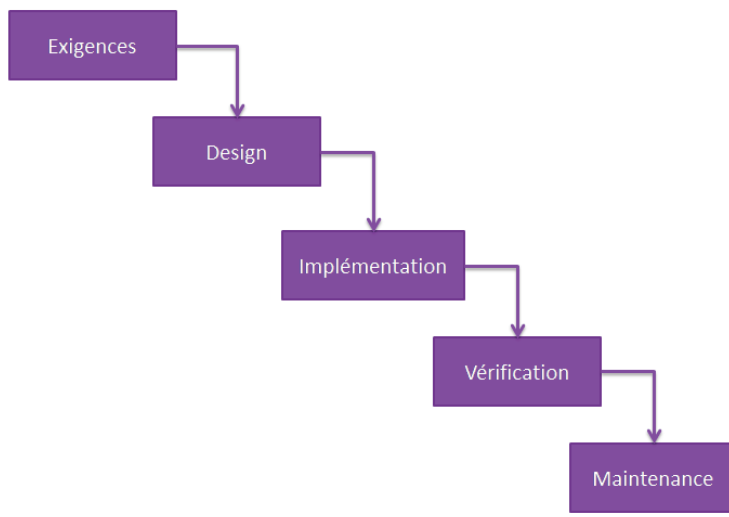


Figure 2 : Modèle Waterfall

Le modèle en cascade présentait de nombreuses limites en termes de réactivité. Aussi dans les années 1980 est apparu un nouveau modèle conceptuel de gestion de projet, le « cycle en V ». Dans ce modèle, les phases de la partie montante doivent renvoyer de l'information sur les phases en vis-à-vis lorsque des défauts sont détectés, ce qui permet de limiter un retour aux étapes précédentes. Par exemple, le document de spécification prérequis pour la conception est également utilisé pour la validation.

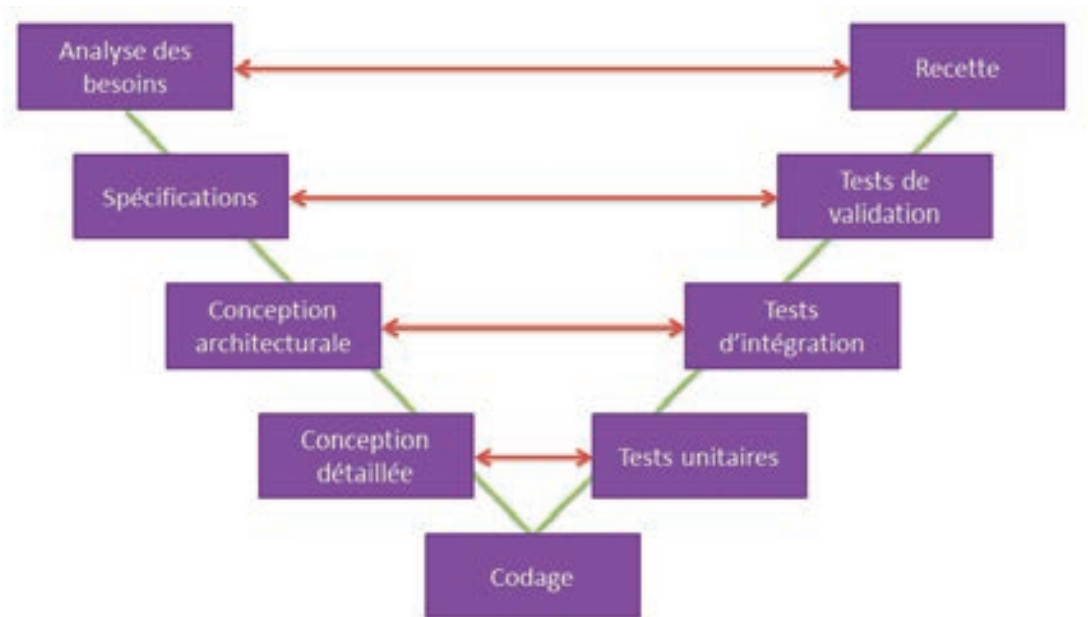


Figure 3 : Cycle en V

En 1988 est apparu le « modèle en spirale » qui reprend les différentes étapes du « cycle en V ». Par l'implémentation de versions successives, cette démarche cyclique permet d'itérer les étapes pour aboutir à un produit de plus en plus complet et dur. Le cycle en spirale met cependant plus l'accent sur la gestion des risques que le cycle en V.



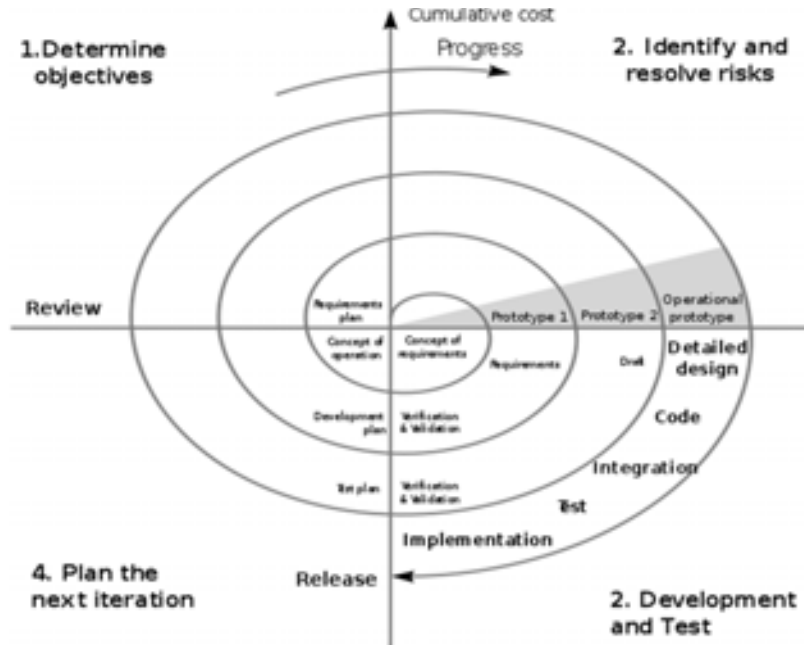


Figure 4 : Modèle en spirale

La méthode MSF (Microsoft Solutions Framework) est une instantiation de ce modèle.



Figure 5 : Méthode MSF

Dans les années 1990 sont apparues les méthodes de « Processus unifié » génériques, itératives et incrémentales ciblant principalement les logiciels orientés objets et dont RUP (Rational Unified Process) est sans doute l'instanciation la plus connue. RUP fait explicitement la distinction entre le cycle de vie, avec des phases et des itérations et les disciplines, comportant la description des rôles, produits et tâches.

## 2. Les méthodes agiles

Dans la même période ont commencé à apparaître des méthodes se voulant plus pragmatiques que les méthodes traditionnelles et impliquant plus le client pour offrir une meilleure réactivité à ses demandes. En 2001, le « Manifeste Agile » formalise la reconnaissance de ces méthodes « agiles » et des pratiques associées sur la base d'un ensemble de 4 valeurs et 12 principes communs. Une des caractéristiques essentielles de l'agilité repose sur la mise en place d'une structure du cycle de développement itérative et incrémentale prenant en compte la collaboration avec le client plutôt que la contractualisation des relations. L'agilité se distingue également en favorisant le travail d'équipe, l'auto-organisation et la responsabilisation.

Les méthodes agiles décomposent les actions en petites étapes de planification (itération de 1 à 4 semaines). Chaque itération cible un travail d'équipe à travers un cycle de développement logiciel complet. Cela permet de minimiser le risque global et de s'adapter rapidement aux changements. L'objectif est d'avoir une version disponible à la fin de chaque itération.

Scrum est sans doute la plus populaire des méthodes agiles. Citons également XP (Extreme Programming), Lean et Crystal Clear.

## 3. Le cas de Scrum

Scrum est une méthode de conduite de projets basée sur des valeurs et principes agiles. Scrum définit un jeu d'activités qui permettent aux clients d'étudier, de guider et d'influencer le travail de l'équipe de développement au cours de sa progression. L'équipe travaille par courtes itérations (appelées aussi sprints) et affine son plan à mesure qu'elle avance dans son travail. Ces différents sprints ont une durée fixe, toujours identique, et fixée à l'origine du projet. Cela permet de connaître la **vélocité** de l'équipe et de pouvoir être toujours plus précis dans l'estimation du temps de réalisation d'une partie de l'application.

### A. La perspective d'une équipe Scrum

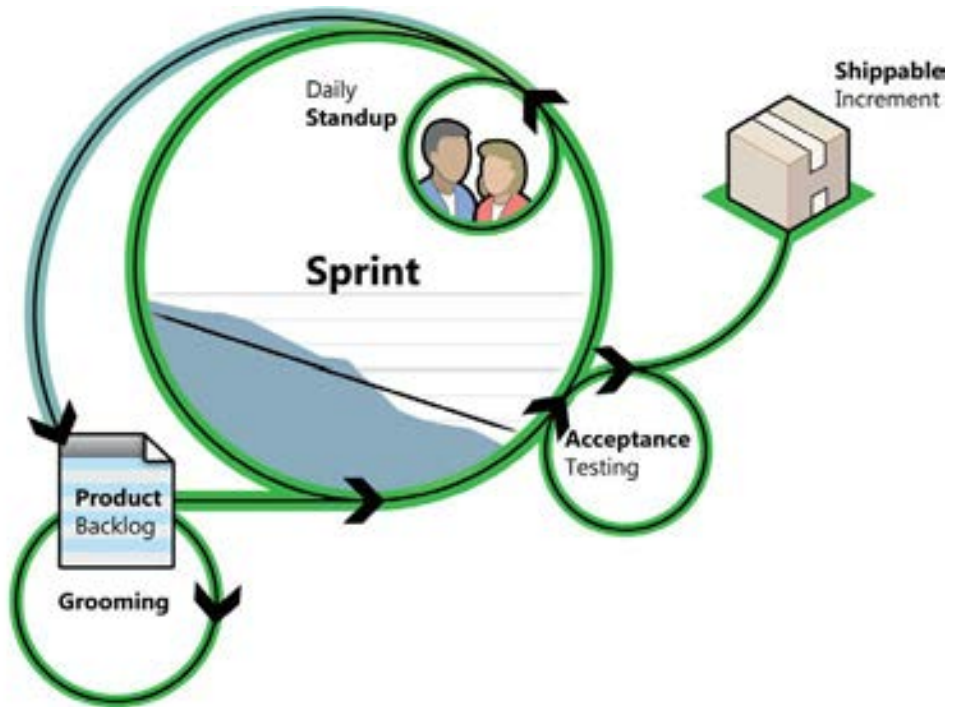


Figure 6 : Scrum, la perspective de l'équipe

Les **choses** à développer sont appelées **stories** en Scrum. Il s'agit de la liste des besoins exprimés par l'utilisateur. Cette liste est maintenue par le **Product Owner**. Avant un sprint, il est décidé de ce que l'équipe s'engage à réaliser. L'équipe participe à ce choix pour :

- S'assurer que tous les membres aient bien compris l'ensemble des stories planifiées
- S'assurer que tous les membres soient d'accord sur le fait que la liste soit réalisable sur la durée d'un sprint

Pendant le sprint, l'équipe s'organise d'elle-même. Les membres choisissent leurs tâches au fil du temps et de leur avancement.

Le chef de projet n'existe pas dans une équipe Scrum, ce rôle n'est pas nécessaire. Un membre de l'équipe joue un rôle appelé **Scrum Master**. Celui-ci n'est pas là pour diriger l'équipe, mais pour s'assurer que la méthode est appliquée comme il faut. C'est l'animateur de l'équipe, le facilitateur qui permet de fluidifier l'avancement du travail.

Lors d'un sprint, l'équipe doit être au courant de ce qui se passe, et chacun doit avoir une vue la plus précise possible du travail effectué par chacun et des problèmes rencontrés. C'est le rôle de la réunion quotidienne appelée **Daily Standup** ou **Daily Scrum**. L'idée n'est pas de

monopoliser l'équipe pendant des heures chaque jour, mais de partager l'information rapidement. Chaque personne s'exprime pendant un temps défini à l'avance, de quelques minutes. Il ou elle indique en quelques mots :

- Ce qu'il ou elle a fait la veille
- Ce qu'il ou elle a prévu de faire le jour même
- Les problèmes qu'il ou elle rencontre si tel est le cas

La dernière étape est très importante car il est probable que cette personne ne soit pas la seule à rencontrer un problème et qu'une solution a peut-être déjà été trouvée par quelqu'un de l'équipe. Au pire, cela permet au Scrum Master d'être au courant et de trouver une solution au problème.

L'objectif de l'équipe est de livrer une nouvelle version de l'application à la fin du sprint. Idéalement, celle-ci contient l'intégralité des stories prévues à l'initialisation du sprint. Si jamais une story n'est pas complètement terminée, elle n'est pas livrée. Elle retourne dans le backlog du produit et pourra être intégrée dans un futur sprint (au mieux, le suivant !).

### ***B. La perspective du Product Owner***

Le **Product Owner** est la personne qui fait le lien entre l'utilisateur final (i.e. la personne qui utilisera l'application) et l'équipe Scrum. Le Product Owner est là principalement pour :

- Récolter les besoins de l'utilisateur, les comprendre et les transcrire sous forme de stories
- Ordonner les stories en fonction de l'importance qu'elle représente pour l'utilisateur

Ces opérations sont réalisées quotidiennement par le Product Owner. Son objectif est de réordonner cette liste constamment pour qu'elle soit le plus proche de la réalité. Une story n'est plus d'actualité ? On l'enlève de la liste ! Il en faut une autre ? On l'ajoute et on la place au niveau correspondant à sa priorité. Une story devient plus importante pour l'utilisateur ? On modifie son placement dans la liste. Etc. !

Avant le sprint, il organise la réunion lors de laquelle les stories du sprint sont choisies. Il doit expliquer chaque story à l'équipe, pour que chacun soit certain d'avoir bien compris de quoi il retourne.

Pendant le sprint, il apporte des détails sur certaines fonctionnalités si nécessaire, et continue d'ordonner le Backlog de produit.

Il est important de comprendre qu'à l'identique du Scrum Master, le Product Owner n'est pas un chef de projet. Il n'affecte pas les tâches aux développeurs et ne les dirige pas. Il ne choisit pas non plus seul la liste des stories qui seront réalisées pendant le sprint. Il le fait en accord avec l'équipe.

A la fin d'un sprint, le Product Owner présente les stories **terminées** aux utilisateurs. C'est un des moments où l'utilisateur a la possibilité de remonter des **feedbacks** pour indiquer ce qui lui plaît, ce qui lui plaît moins et parfois de signaler des dérives vis-à-vis de ses attentes.

Lorsque l'on se trompe de cible pendant un sprint de 2 ou 3 semaines, on ne perd alors **que** 2 ou 3 semaines et pas 6 mois ou 1 an comme c'est parfois le cas pour une organisation en projet plus classique.

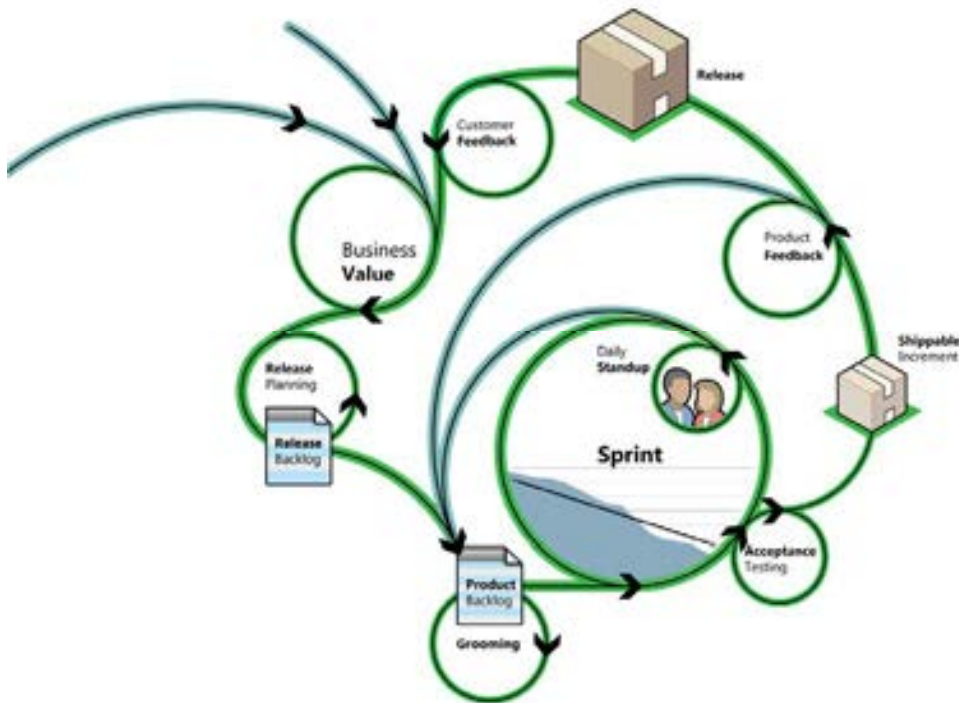


Figure 7 : Scrum, le cycle complet

## Un sprint à travers les outils

### 1. Choisir une méthode de travail

Depuis la version 2005, Team Foundation Server propose aux utilisateurs de définir les éléments de suivi de projet à travers les « éléments de travail » (ou « work items », en anglais). Un élément de travail est une fiche d'information qui n'est ni du code ni de la documentation. Il en existe de plusieurs types et on retrouve communément des « tâches », des « user stories », des « cas de tests » et ainsi de suite. Chaque type de fiche est unique et est défini au sein

d'un modèle de processus. C'est à travers les modèles de processus qu'une équipe peut modifier le comportement de TFS pour lui permettre de suivre la méthode souhaitée.

Pour pouvoir travailler avec TFS, il est nécessaire de créer un projet d'équipe qui est l'espace virtuel au sein duquel est stocké l'ensemble des éléments relatifs au projet. Le modèle de processus défini la création et la configuration d'un projet d'équipe.

Plusieurs modèles sont proposés. Deux sont disponibles par défaut :

- MSF Agile : fortement inspiré de Scrum
- MSF for CMMI : méthode plus classique adaptée à CMMI
- Microsoft Visual Studio Scrum 2.0 : template pur Scrum permettant sa mise en œuvre

Le modèle de processus est donc un moyen efficace de définir la liste des éléments de travail mis à disposition pour l'équipe. Chaque élément de travail est associé à un flux décrivant les différentes étapes liées à son cycle de vie.

Par exemple, on peut imaginer un scénario où le développeur va commencer par implémenter une **tâche**, rattachée à une exigence appelée **Product Backlog** dans la méthodologie Scrum. Une fois le développement de toutes les tâches associées à l'élément du Product Backlog terminé, ce dernier sera soumis à des **cas de test** par l'équipe qualité. Si au cours de ces tests, des anomalies sont détectées, un **Bug** sera alors créé et remonté à l'équipe de développement. Une fois corrigé, le cas de test sera revérifié, et lorsque tous les cas de tests de l'exigence auront été validés avec succès, on fermera alors l'élément du Product Backlog associé et on pourra continuer à se concentrer sur d'autres exigences.

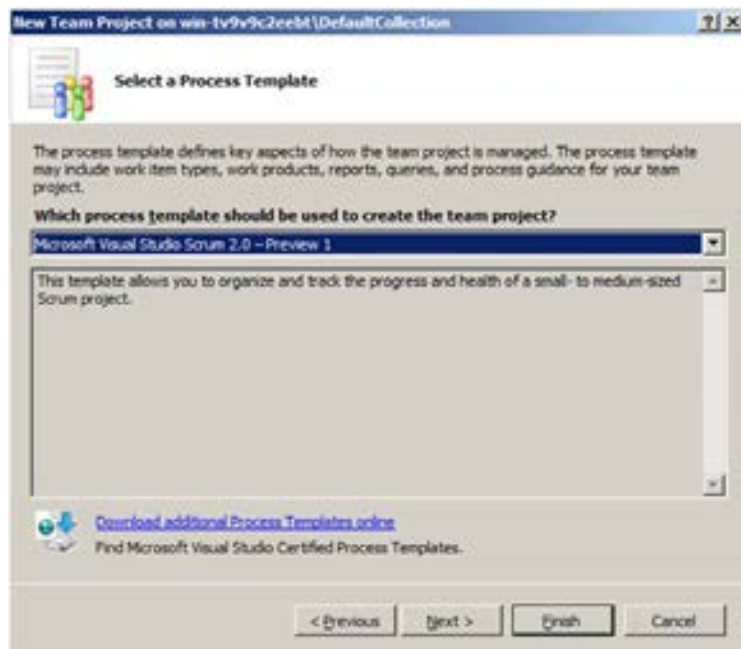


Figure 8 : Création d'un projet TFS

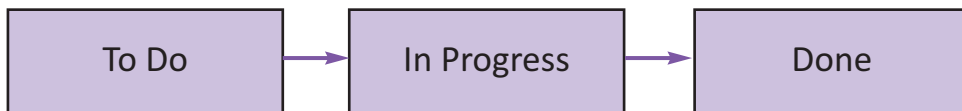
Il est important de comprendre que le choix d'un modèle de processus n'impose pas le respect absolu de l'application d'une méthode de travail. Il s'agit plutôt d'un guide permettant d'accompagner le processus que l'équipe est invitée à suivre, sans contrainte particulière.

La modèle de processus « Microsoft Visual Studio Scrum 2.0 » contient les éléments de travail suivant :

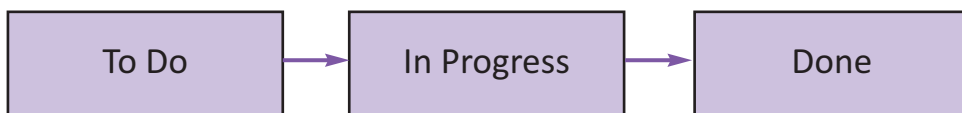
- Product Backlog Item
- Task
- Bug
- Code Review Request
- Code Review Response
- Feedback
- Impediment
- Shared Steps
- Test Case

Dans le flux de travail, ces éléments peuvent avoir différents états.

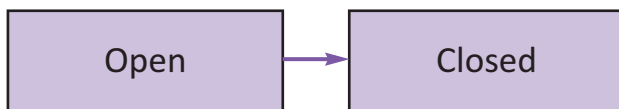
Product Backlog Item et Bug Work Items



Task Work Item



Impediment Work Item



Nous reviendrons sur ces différents éléments par la suite.



## 2. L'équipe

L'équipe est certainement le point le plus important d'une équipe agile. C'est un des axes principaux de Scrum, par exemple. Le projet avance grâce à une équipe, c'est elle qui implémente les fonctionnalités, elle qui réduit le nombre de tâches en les réalisant. L'outil doit donc nécessairement intégrer ce principe.

Jusqu'à présent, il était possible de créer une notion d'équipe dans TFS en exploitant la notion de groupe de sécurité et en créant un groupe par équipe. On mélangeait alors deux aspects différents.

Dans TFS 2012, la notion d'équipe est clairement identifiée. Une équipe dans TFS 2012 n'est plus le simple conteneur d'une liste de personnes et d'un titre. Elle possède des informations complémentaires vitales pour le suivi de projet, à fortiori dans le cadre d'une méthode agile.

On pourra ainsi associer les itérations auxquelles l'équipe participe (nécessaire dans le cadre d'un gros projet, pour distinguer les itérations de chaque équipe), les différents périmètres fonctionnels ou encore le nombre d'heures que pourra consacrer chaque membre, chaque jour, au projet actuel. L'équipe devient donc un élément central dans l'utilisation de TFS 2012.

La version 2010 propose un accès web aux utilisateurs. Celui-ci permet d'avoir accès à la quasi-totalité des fonctionnalités du serveur TFS sans avoir besoin d'installer les outils clients riche. Idéale pour dépanner ou pour donner un accès au suivi de bogues à des utilisateurs externes, cette version légère reste principalement utilisée en appoint de la version riche, le team Explorer.



Figure 9 : Site web de TFS

Dans TFS 2012, la prise en compte de la notion d'équipe se traduit également par une refonte de l'interface web. Les fonctionnalités précédentes sont toujours présentes, mais l'ergono-



mie de l'interface a été entièrement revue. Une partie du site dédiée à l'équipe fait également son apparition. Il s'agit du point central de communication et de gestion du travail de l'équipe. Il permet de visualiser les itérations passées, l'itération courante, de visualiser le backlog global, le backlog d'une itération en particulier et d'avoir une vue sous forme de « post-its ». Ainsi le site web est maintenant placé au centre de la gestion du projet et de l'équipe.

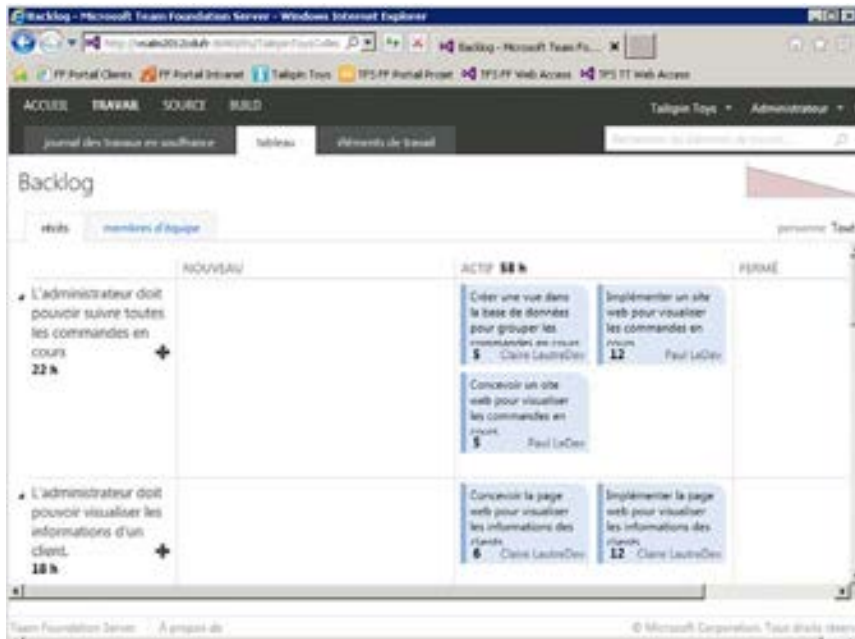


Figure 10 : Vue post-it sur le site web de TFS

### 3. La gestion du backlog



Figure 11 : Scrum, gestion du backlog

Lorsque l'on développe une application, il est toujours nécessaire d'avoir une liste de fonctionnalités à développer. L'organisation de cette liste sera fonction de la méthode adoptée par l'équipe.

- Dans une méthode « classique », la liste des fonctionnalités est découpée selon une arborescence d'exigences.
- Dans une méthode agile, on utilise une liste que l'on nomme **backlog**.

Le backlog est une pile des différentes fonctions proposées par l'application. Cette liste est susceptible d'être alimentée en continu. Elle évolue donc sans cesse et permet de définir les délais de mise à disposition d'une nouvelle fonctionnalité.

Au sein de TFS 2010, ce backlog est alimenté par des éléments de travail dont le type peut varier en fonction de la méthode adoptée :

- Dans le modèle de processus « MSF Agile », ils sont de type **User Story**
- Dans le modèle de processus « Microsoft Visual Studio Scrum 2.0 », ce sont des **Product Backlog Items**. Le concept est globalement le même, il s'agit d'une liste de fonctions à développer pour lesquelles un titre, une description et un état sont indispensables.



Figure 12 : Création d'un product backlog item

Ces fiches permettent ainsi de définir l'intégralité des informations nécessaires à la compréhension de ce qui est attendu pour la fonctionnalité.

Il est possible de créer une **Récit utilisateur** ou un **Élément du backlog** à partir des différentes interfaces proposées à l'utilisateur. Lorsque l'on utilise un outil dans le cadre d'un projet agile, on s'attend à ce que l'usage de cet outil s'inscrive totalement dans le cadre de la démarche adoptée.

De même, les attentes d'un utilisateur peuvent varier en fonction du contexte et de son objectif à un instant t. Par exemple, au démarrage de l'implémentation d'un **Élément du backlog**, un développeur souhaite consulter cet élément sous forme de fiche. En revanche, quand il consulte l'état d'avancement global, il ne souhaite avoir qu'une vue synthétique, idéalement sous forme de tableau à deux dimensions. TFS est là pour permettre d'avoir facilement

toutes ces vues tout en centralisant les données en un seul et unique emplacement.

C'est pour cette raison que les éléments du backlog seront parfois accédés à partir de Visual Studio, parfois à partir de l'interface web, parfois à partir d'Excel (qui est l'un des clients proposés). Peu importe l'interface utilisée, les données sont les mêmes.

Dans TFS 2012, l'interface web est au centre de la collaboration de l'équipe. Presque tous les types d'interactions y sont proposés. La saisie du backlog peut très facilement y être réalisée. Il suffit de saisir le titre du nouvel élément et d'appuyer sur entrée (ou de cliquer sur **Add**) pour que celui-ci apparaisse dans le backlog. Précédemment, seule l'interface Excel permettait une saisie aussi rapide.

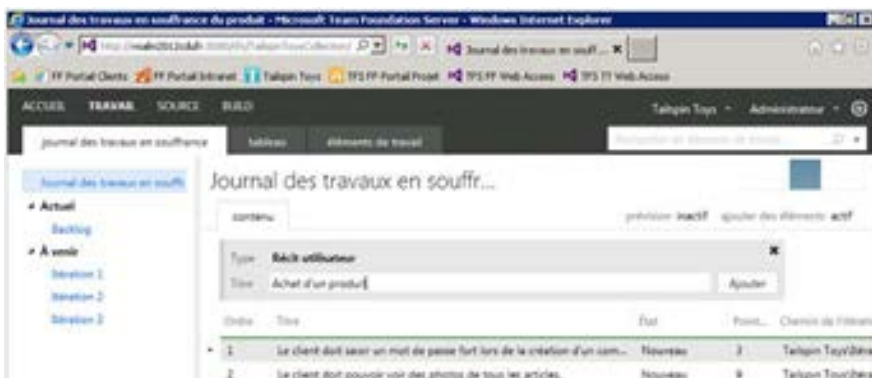


Figure 13 : Visualisation du backlog de produit dans le site web

L'interface présentée ci-dessus est complètement asynchrone ce qui évite de bloquer inutilement l'utilisateur pendant la saisie lors de la sauvegarde de l'élément ajouté.

Un simple glisser/déposer dans l'interface web permet de réordonner les éléments du backlog. Il s'agit d'une action réalisée très fréquemment par la personne en charge de cette liste de fonctionnalités (souvent appelée **Product Owner**). Le gain de temps correspondant est donc fort appréciable.

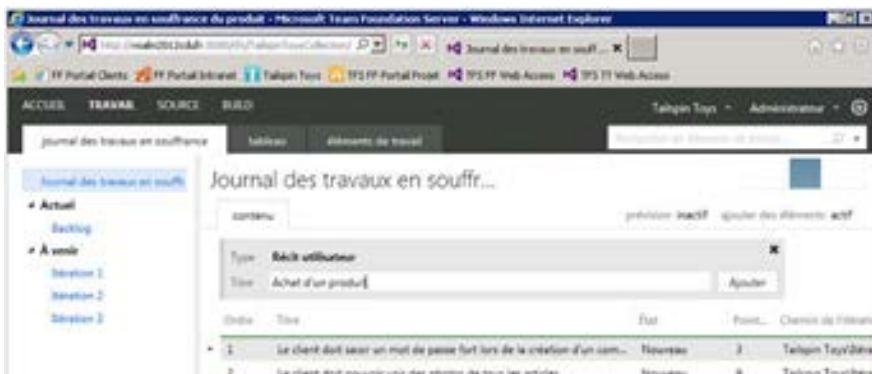
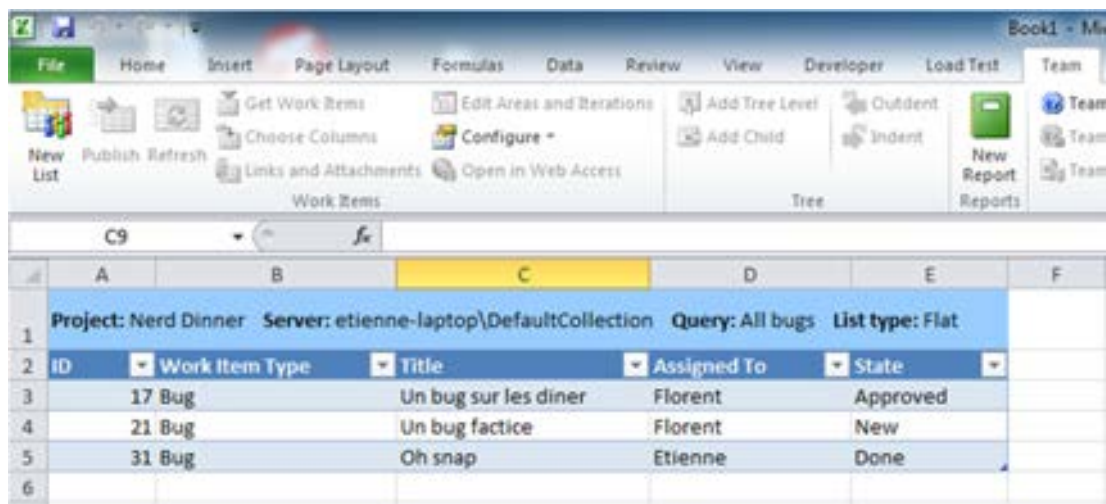


Figure 14 : Reordonner le backlog

L'interface web n'est pas le seul outil à la disposition des utilisateurs pour gérer le backlog. Il est possible d'utiliser Microsoft Excel pour accéder à tous les types d'éléments de travail de Team Foundation Server. Il est ainsi possible de visualiser la liste des bugs, des tâches mais également le backlog du produit et pourquoi pas le backlog du sprint courant avec la liste des éléments du backlog et les tâches qui y sont associées. L'interaction est réalisée de manière asynchrone et il est possible de rafraîchir les données à partir de TFS ou de les mettre à jour en publiant à partir d'Excel.



ID	Work Item Type	Title	Assigned To	State
17	Bug	Un bug sur les diner	Florent	Approved
21	Bug	Un bug factice	Florent	New
31	Bug	Oh snap	Etienne	Done

Figure 15 : Visualisation des éléments de travail avec Excel

## 4. Les storyboards : bien décrire une fonctionnalité

L'équipe qui implémente une fonctionnalité doit avoir une définition claire de ce qu'elle doit faire, que le projet soit piloté en mode « agile » ou pas.

Cela suppose la spécification d'un titre et d'une description détaillée mais parfois, cela ne suffit pas. Cette définition peut être complétée par la fourniture d'éléments complémentaires visuels. A l'heure actuelle, les solutions pour donner rapidement un aperçu d'une application s'articulent autour de logiciels de dessin ou de maquettage comme Microsoft Sketchflow. Une des difficultés rencontrées est alors liée à la non maîtrise de ce type d'outil par l'ensemble des profils concernés.... Cependant, il serait préférable que tout le monde dans l'équipe, du développeur au product owner, puisse donner son avis, exprimer ses idées au niveau de l'interface future de l'application ou d'une fonctionnalité.

Tout le monde connaît PowerPoint. Chacun a déjà créé au moins une fois dans sa vie une présentation en utilisant cet outil de la gamme Microsoft Office. C'est pour cette raison que l'équipe produit en charge de TFS 2012 a choisi cette application comme support à la création de **Storyboards**. L'idée est relativement simple : l'équipe imagine le design des différents

écrans de l'application via PowerPoint, chaque slide de la présentation étant une page d'un site web, un contrôle utilisateur, une interface WPF, et bien d'autres.

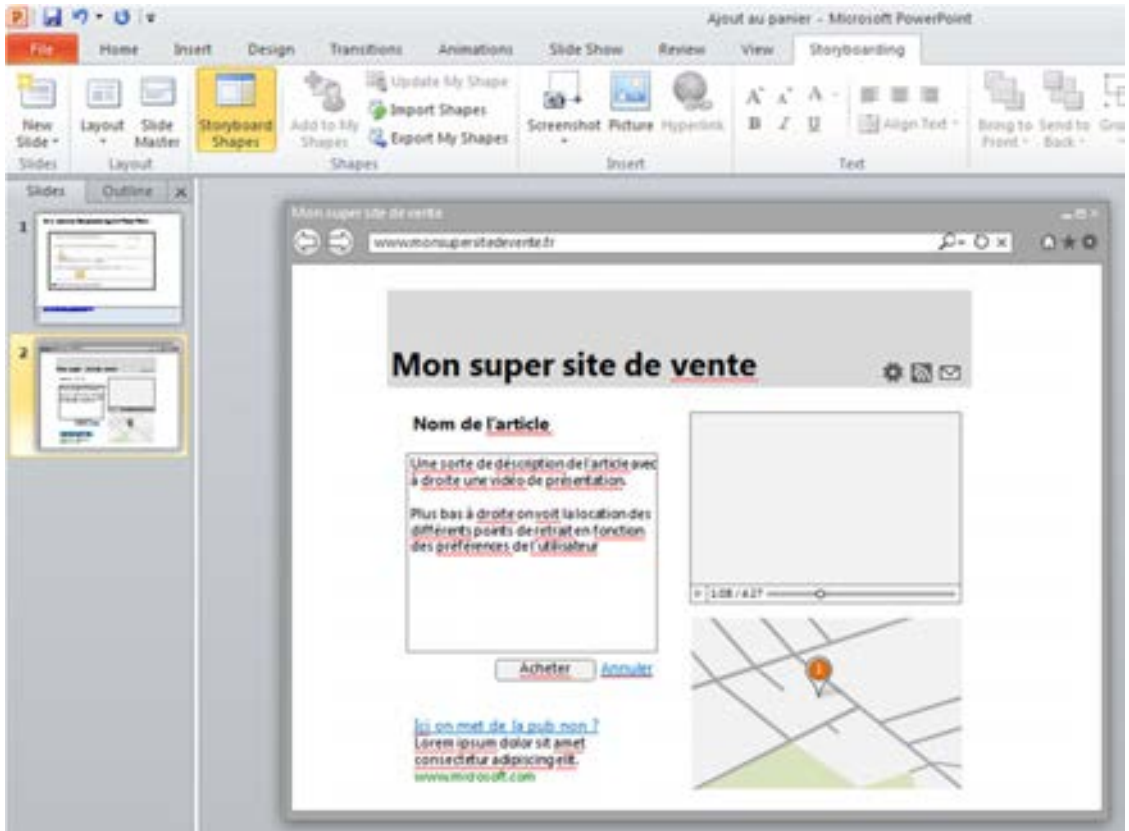


Figure 16 : Créer un storyboard avec le plugin PowerPoint

L'extension **Storyboarding** propose un ensemble de modèles d'application tels qu'une page de site web, un écran Windows Phone 7, une application classique, etc. Ceux-ci permettent de créer de nouveaux slides PowerPoint qui représentent les écrans. Il est alors possible d'enrichir ces écrans avec des formes créées spécifiquement pour le storyboarding. Elles sont utilisées pour ajouter des boutons, des liens, une carte, un exemple de publicité, une grille de données, etc. Plus besoin de prendre des heures pour découper des captures d'écrans pour construire des maquettes, tous les contrôles classiques sont disponibles !

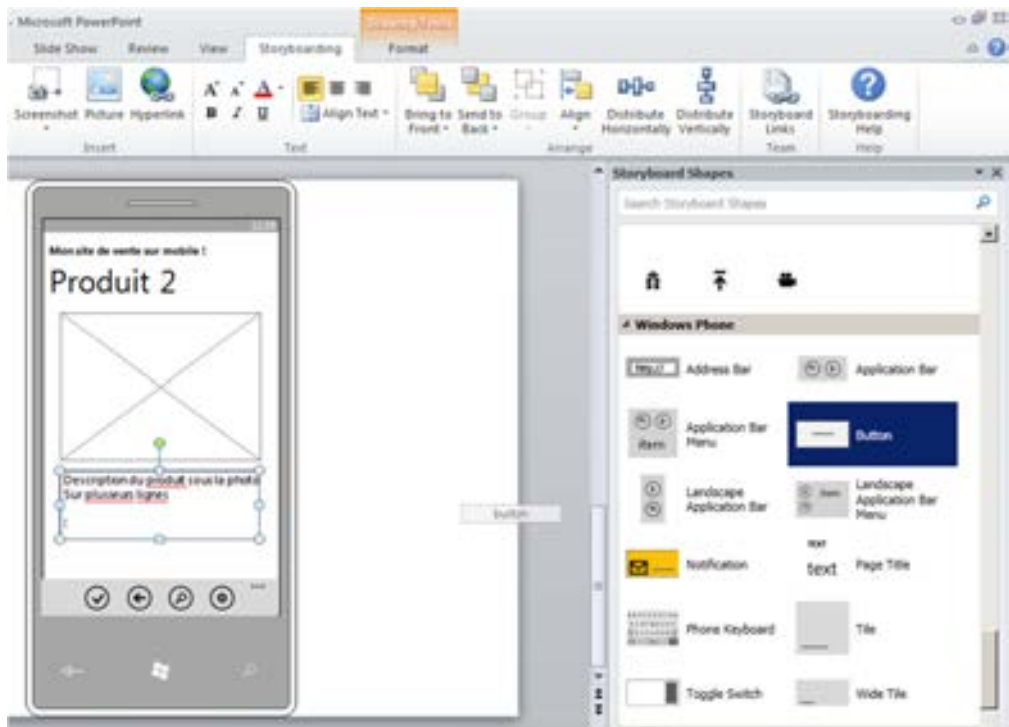


Figure 17 : Construire une maquette avec PowerPoint

PowerPoint présente en outre l'avantage de pouvoir directement démontrer ce que l'on réalisera lors du développement en lançant simplement le mode présentation...

Un storyboard est destiné à décrire un ou plusieurs éléments du backlog. C'est pour cela qu'il est possible de l'associer à un élément de travail de type **Élément du backlog**.

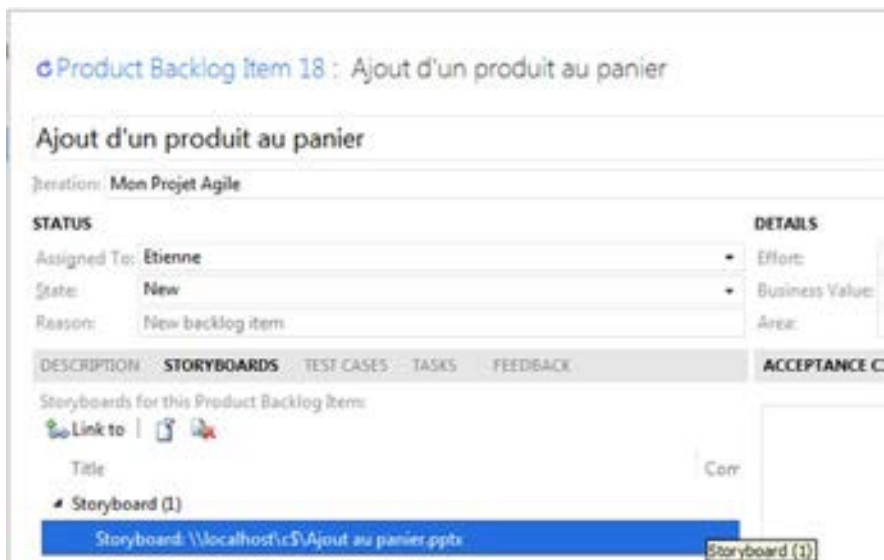


Figure 18 : Associer un storyboard à un élément du backlog

## 5. Planifier un sprint

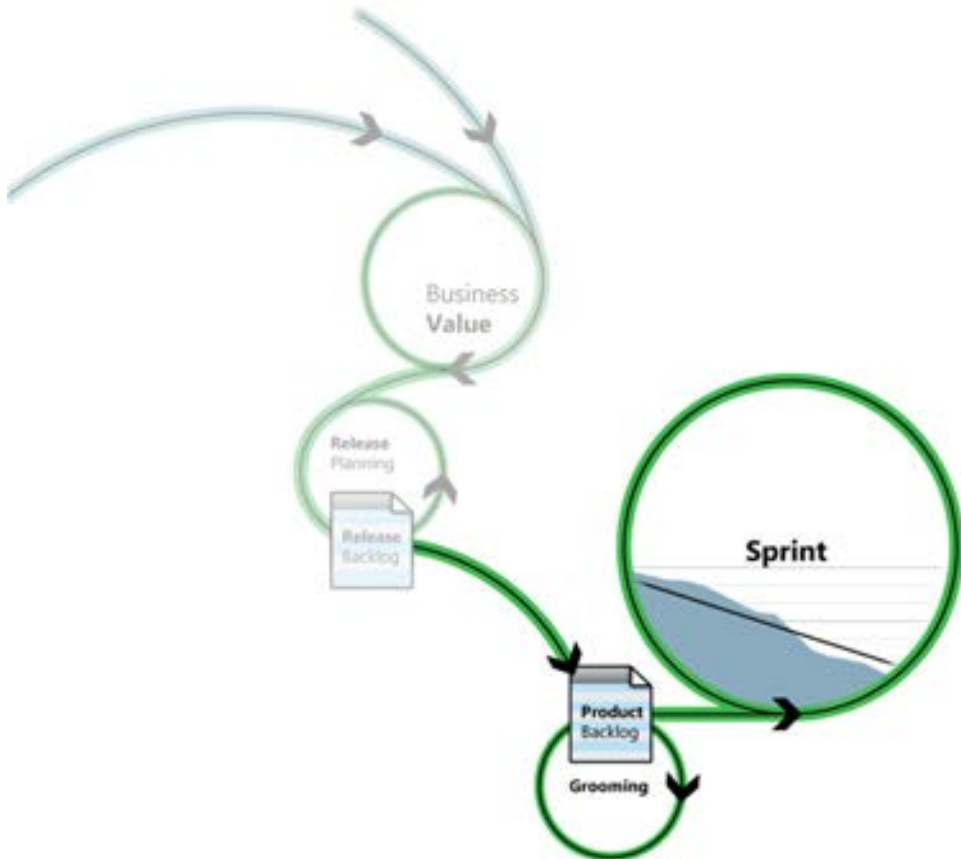


Figure 19 : Scrum, planification

TFS 2012, apporte de nombreuses nouveautés en termes de gestion de projet, et notamment au niveau des itérations puisqu'il est possible désormais de définir les dates de début et de fin de celles-ci. Ceci permet d'afficher automatiquement le bon rapport de burndown par exemple !



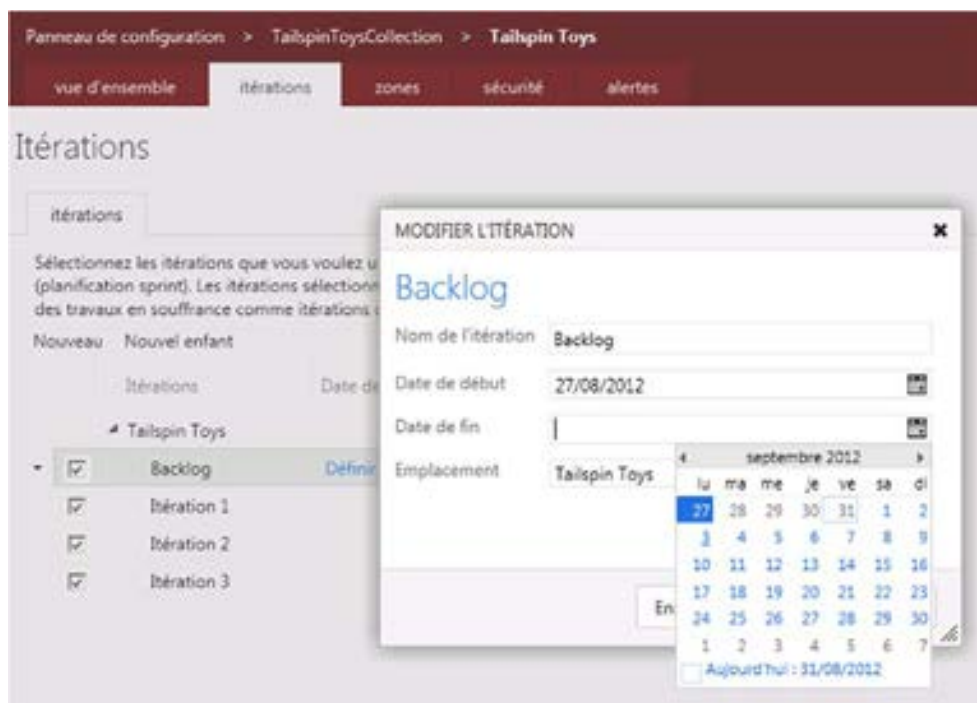


Figure 20 : Assigner une période à un Sprint

A chaque instant, il est possible de filtrer les itérations que l'on souhaite afficher dans le site de l'équipe.



Figure 21 : Consulter liste des Sprint



Le portail de l'équipe permet de visualiser le backlog du produit, ainsi que le backlog de chacune des itérations (ici nommées « Sprint » car nous nous appuyons sur Scrum). L'itération courante est alors mise en évidence à l'opposé des suivantes. Dans les versions précédentes, c'était à l'équipe de savoir où étaient ses éléments de travail pour l'itération courante. C'était à l'équipe de créer les requêtes nécessaires pour filtrer sur une itération. Bref, le suivi du backlog de l'équipe est maintenant largement simplifié.

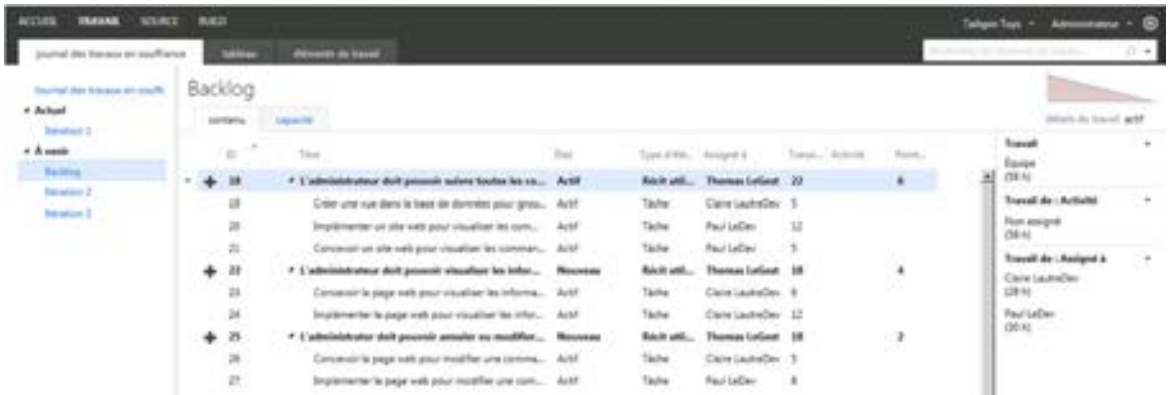


Figure 22 : Visualiser le sprint backlog

Le bouton + cerclé de bleu permet d'ajouter une tâche de réalisation associée à l'élément du backlog. Le temps restant à faire sur chaque tâche est affiché et des informations agrégées sont visibles d'un seul coup d'œil sur la droite de l'écran. On y retrouve le nombre d'heures affectées à chaque membre de l'équipe et le nombre d'heures encore disponibles dans l'itération.

Comme dans la version précédente, une tâche ne correspond pas nécessairement à du développement. Une activité permet de définir le type de travail que la tâche demande d'effectuer. La nouveauté avec TFS 2012, c'est qu'il est possible de définir combien de temps peut consacrer chaque membre de l'équipe pour l'itération en cours, et dans quel type d'activité. En plus de cela, on peut définir les jours d'absence de chacun (les congés par exemple).

## Itération 1

contenu capacité

Membre d'équipe	Capacité par jour	Activité	Jours de congé
Administrateur	0		0 jours +
Claire LautreDev	4	Développement	0 jours +
Marie LesTests	6	Spécifications	0 jours +
Paul LeDev	0		0 jours +
Thomas LeGest	0		0 jours +
Jours de congé de l'équipe			0 jours +

Enregistrer les modifications Annuler les modifications

Figure 23 : Planifier la capacité pour le sprint

L'onglet **tableau** du site d'équipe donne accès à une visualisation sous forme de tableau bien connu des utilisateurs de Scrum. Dans la vue **Élément du backlog**, les tâches d'avancement sont représentées par élément du backlog du produit.

## Itération 1

récits membres d'équipe

	NOUVEAU	ACTIF
Le client doit pouvoir créer un nouveau compte. +	<div>Tache 1</div> <div>Administrateur</div>	
Le client doit pouvoir se connecter au site avec l'identifiant du compte. +		<div>Tache 2</div> <div>Administrateur</div>

Figure 24 : Visualiser le sprint sous forme de post it

La vue **Membres d'équipe**, quant à elle, affiche ce board pour chaque utilisateur.

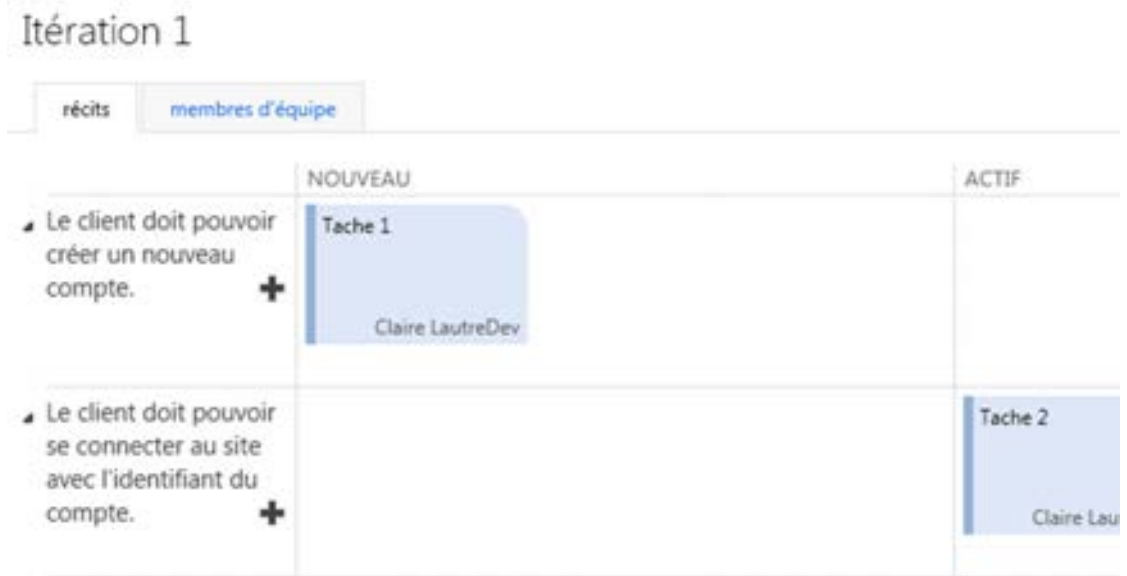


Figure 25 : Visualiser le sprint sous forme de post it par membre

**Note :** Chaque colonne correspond à un état dans TFS. Si un utilisateur modifie cet état avec un autre outil, la modification sera évidemment visible dans cette interface. Il ne s'agit que d'une nouvelle vue sur les éléments de travail.

## 6. Le nouveau flux de travail du développeur

Le site web n'est pas la seule fonctionnalité de TFS 2012 qui a été entièrement revue. Le client riche sous forme d'onglet dans Visual Studio : **Team Explorer** voit également son ergonomie retouchée. La navigation à travers les fonctionnalités est très différente des versions précédentes, notamment avec l'apparition de **Mon travail** qui présente au développeur son travail. Il y retrouve ce qu'il est en train de réaliser, ce qu'il va réaliser, et ce qu'il a mis de côté pour le moment.

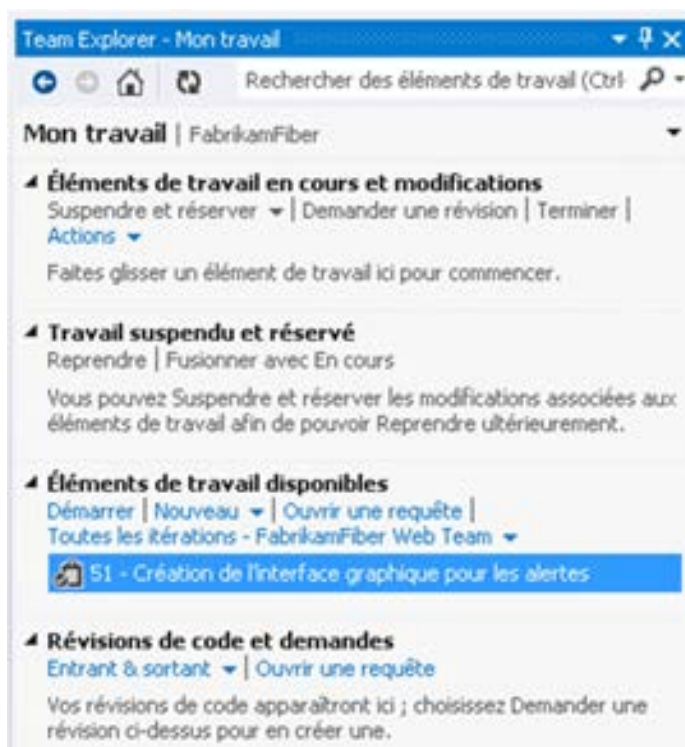


Figure 26 : Team Explorer, page d'accueil

Un tableau de bord est affiché et permet de choisir les tâches sur lesquelles on souhaite travailler.



Figure 27 : Team Explorer, Mon travail

Lorsque le développeur souhaite commencer à travailler sur une tâche, il l'ajoute au travail en cours de réalisation. Ceci est une différence majeure en terme de logique d'utilisation de TFS par rapport à la version précédente: l'utilisateur indique ce qu'il veut faire au moment où il débute, et non pas au moment où il a terminé (lors de l'archivage). Ceci est une approche beaucoup plus logique.

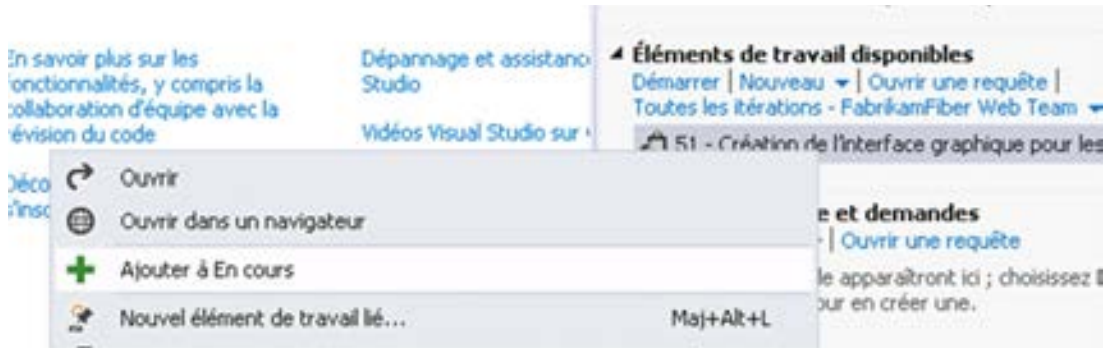


Figure 28 : Team Explorer, démarrer une tâche

Après avoir réalisé les modifications dans le code source que le développeur estime nécessaires à la réalisation de la tâche qu'il a choisie, il s'apprête à archiver ses modifications. La fenêtre **Mon Travail** affiche alors un résumé des modifications réalisées :

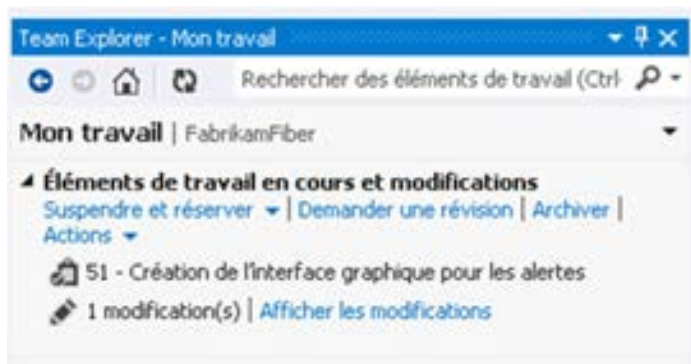


Figure 29 : Team Explorer, visualiser son travail en cours

En cliquant sur **Afficher les modifications** le développeur peut visualiser le détail de ses modifications.

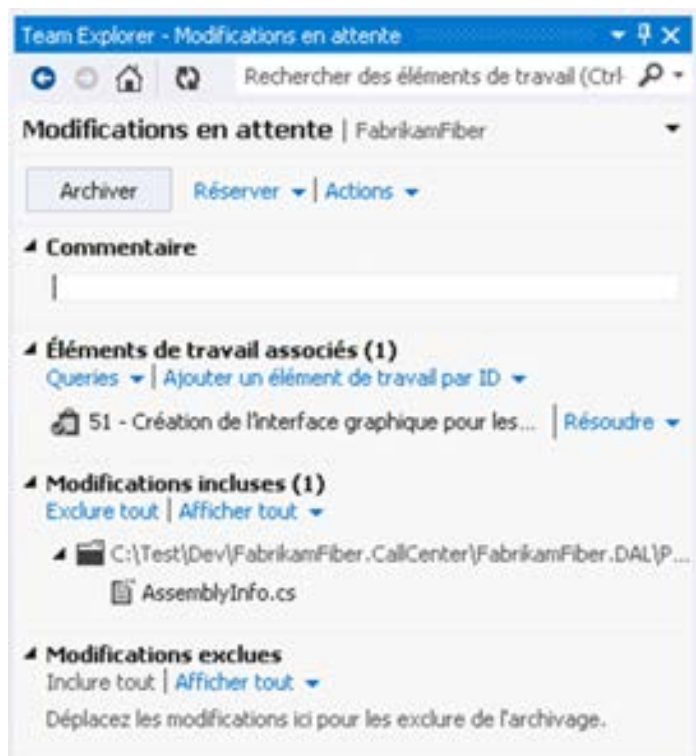


Figure 30 : Team Explorer, modifications en attentes

L'équipe a décidé de faire relire le code de chacun par un autre membre de l'équipe, pour éviter les fautes d'inattention. De ce fait, plutôt que de réaliser immédiatement un checkin, le développeur décide de demander à l'un d'entre eux, d'effectuer une relecture. Pour cela, il utilise la fonctionnalité de demande de **révision** :

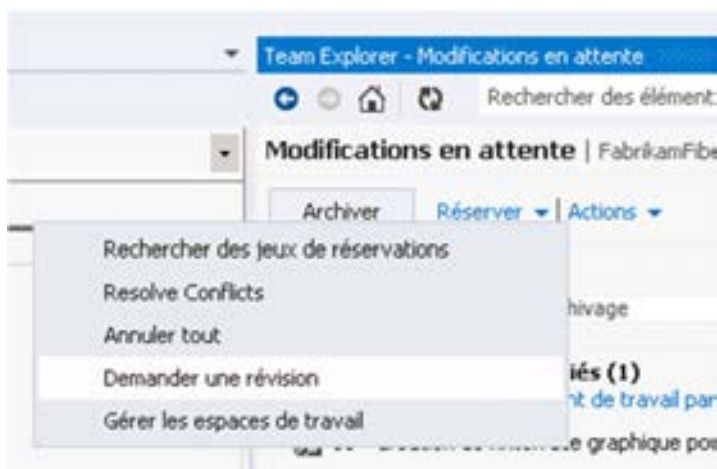


Figure 31 : Team Explorer, demander une relecture de code

Techniquement, ceci génère un élément de travail de type **Demande de révision** mais ceci est complètement transparent pour le développeur qui n'a qu'à préciser le nom du relecteur qui lui semble le plus adapté et préciser sa demande avec un commentaire.

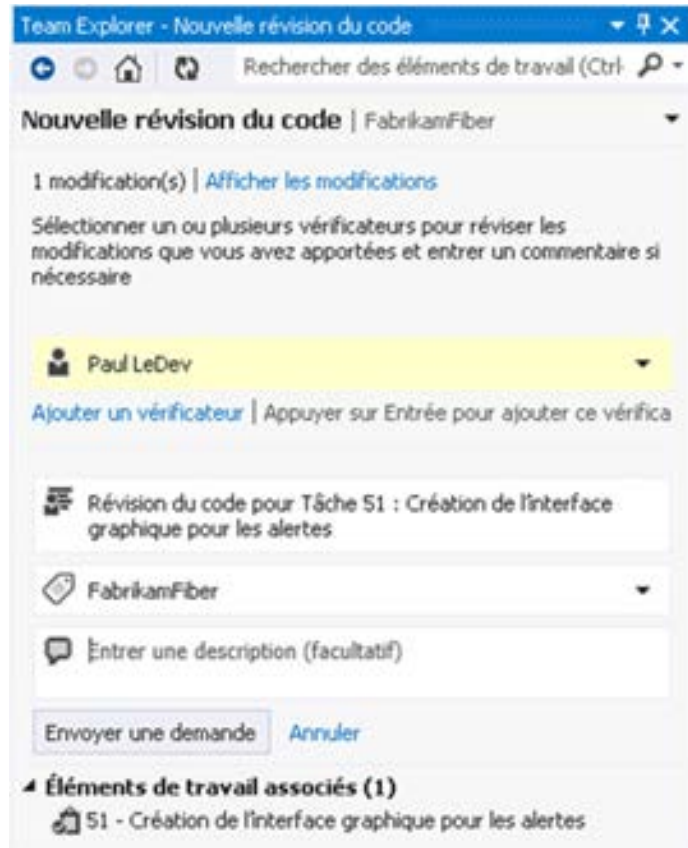


Figure 32 : Team Explorer, préciser la relecture de code

Il n'est pas nécessaire de prévenir le développeur destinataire de la demande, car il la voit instantanément apparaître dans sa page **Mon travail**.



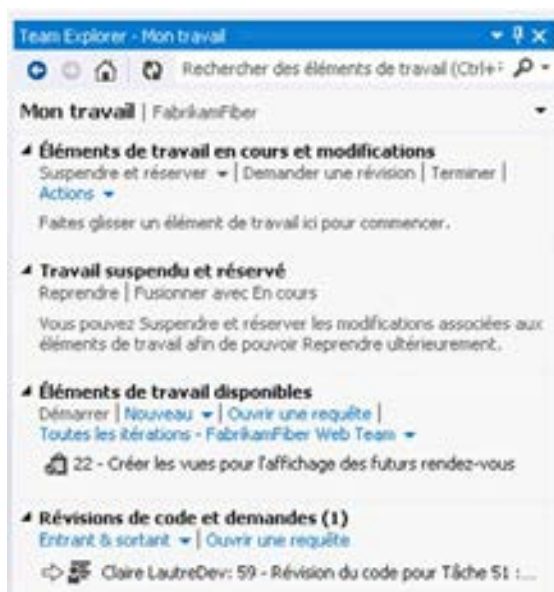


Figure 33 : Team Explorer, visualiser la liste des demandes de relecture

Lorsqu'il consulte la demande, le relecteur visualise le contenu détaillé. Il a notamment accès à la tâche qui est à l'origine du travail, à la liste des relecteurs s'il n'est pas le seul, ainsi qu'à la liste des fichiers modifiés.



Figure 34 : Team Explorer, interface de suivi de relecture de code



En cliquant sur un fichier, Visual Studio présente au relecteur les différences apportées au fichier. Pour cela, il s'appuie sur le nouvel outil de comparaison et de fusion de Visual Studio 2012.

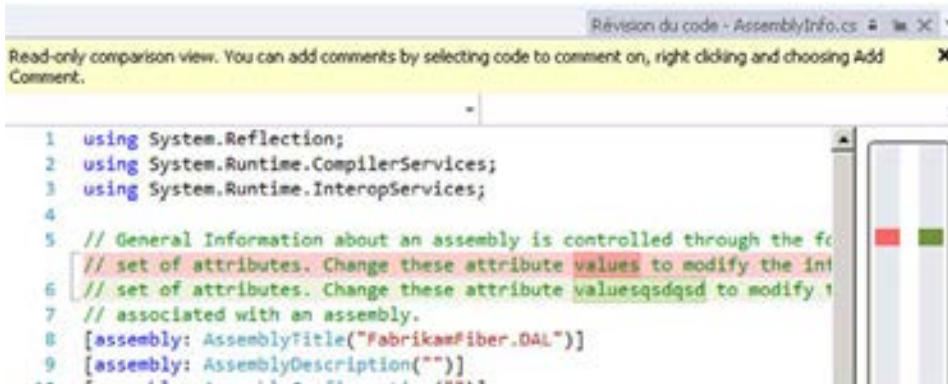


Figure 35 : Nouveau comparateur de code de Visual Studio

La puissance de l'outil de relecture dans TFS 2012 lui permet alors de préciser un commentaire pour une sélection de texte dans le code, sur le même principe qu'une relecture dans Microsoft Word !

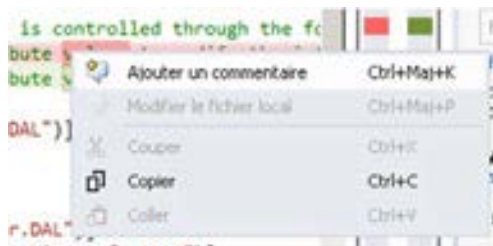


Figure 36 : Ajouter un commentaire de relecture dans le code

Le commentaire apparaît alors dans le Team Explorer de la personne à l'origine de la relecture.



Figure 37 : Ajouter un commentaire sur un fichier pendant une relecture

Si nécessaire le développeur et son relecteur itèrent plusieurs fois jusqu'à obtenir une version qui convient et le relecteur accepte la version.



Figure 38 : Finaliser une relecture de code

Le développeur à l'origine de la demande est au courant de la validation et peut à son tour finaliser la relecture :

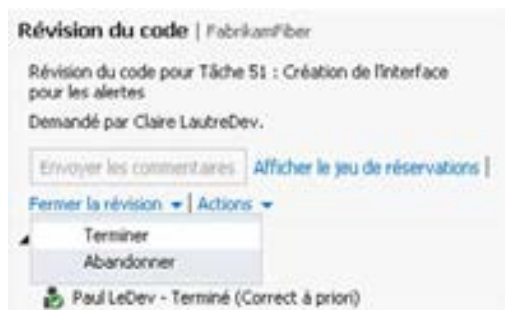


Figure 39 : Valider une relecture de code

Il ne reste plus alors qu'à faire un archivage en cliquant sur Checkin. On remarque que la tâche d'origine est marquée comme résolue par l'association mais également que le work item de Révision de code est associé à l'archivage.

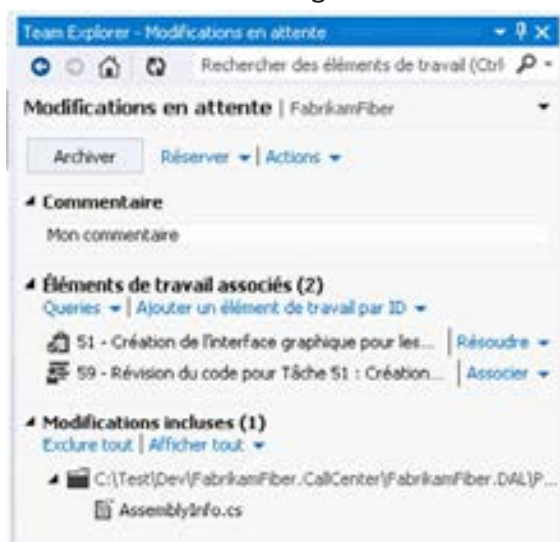


Figure 40 : Partager le code avec le reste de l'équipe

## 7. Vous avez dit tests unitaires ?

Toujours dans l'esprit de fournir du code de qualité, les tests unitaires, bien trop souvent oubliés, sont un moyen efficace de vérifier rapidement les nombreuses fonctionnalités implémentées. On décrit généralement les tests unitaires comme étant « du code testant du code ». Ceux-ci seront écrits dans un langage .NET (C#, Visual Basic ou C++ pour ne citer qu'eux) et vont mettre en œuvre d'autres morceaux de codes compilés, sous forme de .dll ou d'exécutables, tout en contrôlant le résultat obtenu en sortie. (Exemple : Une méthode de test qui s'appellerait **AdditionTest** et qui utiliserait la méthode **Addition** d'un exécutable **Calculatrice.exe**, avec comme paramètre **1** et **2**, et qui attendrait comme résultat **3**).

L'écriture d'un test unitaire permet également de fournir un exemple d'utilisation pertinent du morceau de code testé. Des tests unitaires bien commentés sont bien souvent un moyen efficace de comprendre l'utilisation d'une méthode particulière.

L'aspect « vérification d'une fonctionnalité » des tests unitaires a par ailleurs donné naissance à une méthode de développement appelée **Test Driven Development**, abrégée TDD. Celle-ci préconise de commencer par décrire les comportements attendus dans des tests unitaires, comportements qui ne sont pour l'instant pas encore implémentés, puis de développer l'application testée pour répondre au comportement décrit et ainsi permettre le succès des tests unitaires.

L'importance des tests unitaires n'est plus à démontrer, c'est pourquoi la nouvelle version de Visual Studio va également venir renforcer cet aspect. Au programme, une nouvelle interface appelée Unit Test Explorer et le support d'autres Framework de tests unitaires tels que NUnit ou encore XUnit.



Figure 41 : Unit Test Explorer

Lorsque l'on décide d'utiliser des tests unitaires durant le développement d'une solution, plusieurs Framework sont disponibles. On citera bien entendu le Framework intégré à Visual Studio, mais il faut également en citer d'autres qui ont déjà fait leurs preuves tels que NUnit ou bien XUnit. S'il est conseillé de préférer l'utilisation du Framework intégré à Visual Studio, il existe de nombreux scénarios pour lesquels il est important de pouvoir également gérer les Framework tiers de tests unitaires, ne serait-ce que pour réutiliser des tests déjà existants

ou tout simplement parce que certains développeurs sont plus habitués à ces Framework tiers. À partir de cette nouvelle version, une interface est maintenant disponible pour pouvoir intégrer n'importe quel Framework dans la fenêtre du **Unit Test Explorer**, et ainsi pouvoir gérer d'un seul endroit tous les tests unitaires, provenant de n'importe quel Framework.

Liées à l'exécution de ces tests unitaires en batterie, Visual studio 2010 propose comme dans les versions précédentes des fonctions de couverture de code et permet maintenant de gérer les tests impactés.

Tout au long du développement de l'application, il faut veiller à ce que la valeur de couverture de code soit la plus élevée possible pour garantir que la majorité des lignes de codes de l'application soient soumises à une vérification unitaire. La couverture de code est obtenue à l'aide des résultats des différents tests unitaires qui sont exécutés. En effet chaque test unitaire va solliciter certaines lignes de code spécifique à l'application testée. Dans le cas du test de la méthode d'addition, la méthode de test **AdditionTest** va tester uniquement les lignes de codes contenues dans cette méthode. Si toutes les lignes de codes contenues dans la méthode d'addition sont sollicitées lors de son test, alors la couverture de code sera de 100% pour cette méthode. À l'inverse, dans le cas d'une méthode de division par exemple, un simple test unitaire sur cette méthode permettra de tester le fonctionnement d'une division normale, mais ne permettra pas forcément de tester les lignes de codes spécifiques à la division par zéro (qui souvent lèveront une exception). La couverture de code sera, dans ce cas précis, inférieure à 100%. La somme des couvertures de code de chaque méthode permettra de définir la couverture de code de la classe contenant lesdites méthodes. Là encore, un pourcentage sera donné comme étant la couverture de code de celle-ci.

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
Administrator@VSALM 2012-01-09 13:22:42	1232	97.86 %	27	2.14 %
FabrikamFiber.Web.dll	1232	97.86 %	27	2.14 %
FabrikamFiber.Web	549	100.00 %	0	0.00 %
FabrikamFiber.Web.App_Start	28	100.00 %	0	0.00 %
FabrikamFiber.Web.Controllers	636	95.93 %	27	4.07 %
CustomerController	59	41.30 %	27	58.70 %
Create()	3	100.00 %	0	0.00 %
Create(FabrikamFiber.DAL.Models.Customer)	2	20.00 %	8	80.00 %
CustomerController(FabrikamFiber.DAL.Data.Cus...	0	0.00 %	2	100.00 %
Delete(int)	0	0.00 %	0	0.00 %
DeleteConfirmed(int)	0	0.00 %	3	100.00 %
Details(int)	4	100.00 %	0	0.00 %
Edit(FabrikamFiber.DAL.Models.Customer)	2	20.00 %	8	80.00 %
Edit(int)	4	100.00 %	0	0.00 %
Index()	4	100.00 %	0	0.00 %
EmployeeController	46	100.00 %	0	0.00 %
HomeController	29	100.00 %	0	0.00 %

Figure 42 : Couverture de code

Au fur et à mesure que l'application développée évolue, de plus en plus de tests unitaires sont créés, et il peut alors devenir de plus en plus fastidieux et coûteux en temps de tous les exécuter. C'est pour cela qu'il serait plus pertinent de n'exécuter que les tests liés aux

fonctionnalités qui ont été développées : les tests impactés permettent de vous aider à faire ce choix. Les tests impactés ne prennent tout leur sens qu'après la première exécution de tests unitaires. À partir de celle-ci, et lors de chaque compilation de l'application, la fenêtre des tests impactés vous permettra de visualiser les tests unitaires suggérés par Visual Studio qui ont pu être potentiellement invalidés par les modifications effectuées. Si l'on prend par exemple la méthode de division, un premier test unitaire sera développé afin de vérifier le comportement normal de cette méthode (*En passant comme paramètre 4 divisé par 2, on obtient en résultat 2*). Si maintenant on décide de modifier cette même méthode pour implémenter la division par zéro, il semble évident qu'il faudra rejouer le premier test unitaire vérifiant son comportement normal et c'est ici ce que Visual Studio interviendra en vous suggérant ce test unitaire via la fenêtre des tests impactés.

## 8. Le besoin d'intégration continue

Un des principaux objectifs de l'industrialisation des projets de développement est d'améliorer la qualité des solutions produites par nos équipes. Dans le cas de projets pouvant durer plusieurs mois, il paraît évident que vérifier cette qualité en fin de chaîne, une fois tout le travail effectué, est un risque important. Que faire si rien ne fonctionne ? Sommes-nous certains que les différentes briques s'intègrent correctement les unes aux autres ? Plus simplement: est-ce que le code contenu dans le référentiel du contrôle de code source compile ? Est-ce que tous les tests unitaires sont validés ?

Il est facile d'admettre que découvrir ce type de problèmes en fin de cycle, alors que l'on approche de la fin du projet, engendre dans la majorité des cas :

- du stress
- des corrections rapides et parfois hasardeuses
- un décalage de la livraison du projet
- un échec complet dans certains cas plus rares

... et tout ceci résulte en une qualité moyenne, voire médiocre du résultat produit.

L'intégration continue permet d'éviter de se retrouver dans cette situation. Il s'agit d'une pratique qui vise à inciter les développeurs à partager le plus fréquemment possible leur code, au minimum une fois par jour. Dans le cadre de Team Foundation Server, ce partage s'effectue bien évidemment à travers le contrôle de source. En outre, le concept d'intégration continue nécessite une vérification d'un certain nombre de critères de qualité du travail partagé par le développeur. Cette opération est automatisée pour optimiser le travail de l'équipe à l'aide de ce que l'on appelle couramment un système de « build ». Au sein de la gamme de produits Visual Studio ALM, celui-ci est appelé « Team Build ».

Ce système d'intégration continue doit être capable d'effectuer un ensemble d'opérations

permettant d'obtenir un indicateur clair de la qualité du code actuellement présent dans le contrôle de source. Les opérations les plus courantes réalisées lors de ce type de processus sont :

- la compilation de la dernière version du code
- l'exécution d'outils de qualité (analyse statique de code, par exemple)
- l'exécution de tests unitaires
- l'exécution de tests de non régression fonctionnelle automatisés
- l'obtention du pourcentage de lignes de code couvertes par les tests exécutés

Le résultat de ces différentes actions donne lieu à la construction d'un rapport permettant de savoir où l'on en est. Si les résultats sont positifs, on a de bonnes raisons d'être confiant. Si quelques indicateurs sont au rouge, il faut comprendre pourquoi et redresser la barre dans la bonne direction. Si au contraire tout est au rouge et que l'application ne compile pas, toute l'équipe doit se sentir concernée et effectuer les actions nécessaires à la correction des problèmes identifiés.

Ce processus permet de vérifier la qualité tôt et souvent. Il est en général beaucoup plus facile de corriger les problèmes au fil de l'eau que d'un seul coup en fin de projet.

Ces processus permettent d'effectuer des opérations automatiques supplémentaires telles que la livraison automatique de l'application au sein d'un environnement d'intégration. Ceci permet à l'équipe de test d'avoir une version toujours à jour de l'application et implicitement de détecter les anomalies rapidement.

### Notion de définition de build

Les flux de travail sont des modèles et ne peuvent pas être utilisés et exécutés tels quel. Il est nécessaire de les exploiter au sein de **définitions de build**. Il s'agit d'instances de ces flux de travail potentiellement configurées différemment. Si votre projet d'équipe TFS contient deux solutions Visual Studio, il est tout à fait possible de créer une définition de build pour chacune d'entre elles.

L'exécution d'un processus d'intégration continue, revient en réalité à la mise en file d'attente de l'exécution d'une des définitions de build d'un projet d'équipe TFS.

C'est au sein de ces définitions de builds que vous pouvez choisir les différentes actions à réaliser lors de l'exécution du processus :

- La version du code à utiliser
- La ou les solutions à compiler et dans quelle(s) configuration(s)
- Les tests à exécuter (si activé)
- Le nom à donner à la version générée à chaque exécution
- Et ainsi de suite...



## Quand s'exécute la build?

Team Build propose un système de déclencheur avancé permettant de choisir finement le moment où le processus sera exécuté.

Il en existe 4 principaux :

- **Manuel** : L'exécution est toujours effectuée manuellement par un utilisateur.
- **Intégration continue** : Une exécution est lancée chaque fois qu'un développeur archive du code dans le contrôle de source.
- **Archivage contrôlé** (Gated Checkin) : Le code que le développeur archive est placé dans un espace privé. La build est alors exécutée en prenant la dernière version de tous les fichiers. Le code n'est réellement archivé que si la build est un succès. Sinon la personne est avertie qu'il reste du travail pour que les modifications soient acceptées.
- **Planifié** : Pour certaines équipes, l'intégration lors de chaque archivage peut être inutile. Cette option permet ainsi de planifier une exécution par jour par exemple si cela suffit.

## Rapport et résultat de build



Figure 43 : Rapport de build d'intégration continue

Chaque exécution de build donne lieu à l'attribution d'un **numéro**. Celui-ci permet d'identifier la version intermédiaire créée lors de l'exécution. Lors de chaque build, le numéro est utilisé pour **appliquer une étiquette** dans le contrôle de source. Ceci permet par la suite de pouvoir identifier et télécharger la version exacte qui a été utilisée par le système pour compiler cette version intermédiaire même si le code a été modifié depuis. Le numéro de build

est également utilisé pour nommer un **répertoire partagé** dans lequel le processus a déposé le résultat et les logs de la compilation ainsi que les résultats de tests.

## 9. Test Manager 2012, un outil agile

La validation de la qualité d'une application ne s'arrête pas à la gestion des tests techniques. Il est important de valider unitairement chaque composant ou d'optimiser les performances globales mais rien ne remplace la validation fonctionnelle. Un utilisateur doit utiliser l'application pour détecter les incohérences graphiques ou métiers, donner son ressenti sur l'ergonomie et identifier les anomalies liées à l'intégration globale.

Ce rôle de validation fonctionnelle est pris en charge par l'équipe de test, recette ou qualité. Contrairement aux limitations d'autres solutions du marché, Visual Studio 2010 apporte une cohérence générale entre tous les métiers tournant autour du développement du projet. Les testeurs fonctionnels sont complètement intégrés au processus et un outil leur est dédié pour effectuer leur travail et communiquer avec le reste de l'équipe. Bien qu'inclus dans la gamme de produit Visual Studio, Microsoft Test Manager n'est pas lié à l'outil de développement Visual Studio et son ergonomie est radicalement différente. Il est conçu pour optimiser le travail de l'équipe qualité et est connecté à la plateforme de collaboration Team Foundation Server pour l'échange d'information et la centralisation de données.

Au-delà d'un simple outil de gestion de cas de tests, Test Manager est là pour accompagner les équipes de testeurs fonctionnels. Tout est fait pour faciliter la communication avec le reste de l'équipe et notamment avec les développeurs. L'outil d'exécution de tests manuels permet d'apporter un grand nombre de détails pour chaque étape tels que des commentaires ou des captures d'écrans. Un système évolué de collecte d'informations permet de regrouper une quantité importante d'éléments qui sont automatiquement attachés à toute nouvelle anomalie reportée via l'outil. Dès qu'une anomalie est corrigée par un développeur, l'équipe qualité est avertie et est invitée à ré-exécuter le test qui avait permis de mettre en évidence le problème. De même, lorsqu'une nouvelle version de l'application est disponible, une liste de tests de non-régression de premier niveau peut être automatiquement générée, évitant un long travail d'analyse des modifications apportées.

Un premier niveau d'automatisation est possible directement à partir de l'outil pour permettre à un testeur fonctionnel non technique d'effectuer des enregistrements de tests, de les rejouer et de les transmettre aux équipes de développeurs ou de testeurs techniques comme base d'une automatisation plus étendue.



L'outil est découpé en deux ensembles de fonctionnalités différents. Le **Centre de tests** contient tous les outils relatifs à la mise en place de plans de test, la gestion de cas de test, leur exécution et leur suivi. Le **Centre lab** permet quant à lui la manipulation des environnements d'exécution et surtout la gestion des machines du **Lab Management**, détaillée dans un des chapitres suivants.

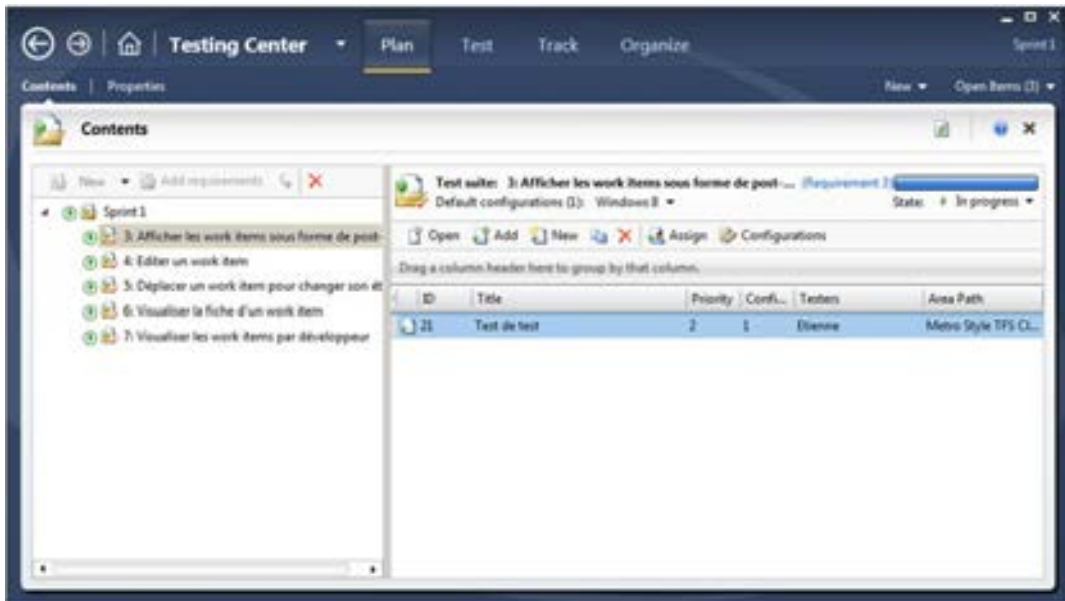


Figure 44 : Test manager

La version 2010 marque l'arrivée de cet outil et sa mise en place a permis à un grand nombre d'équipes d'améliorer leurs processus de qualité en facilitant la communication entre les équipes de tests et de développement.

Avec Test Manager 2012, il n'est désormais plus impératif de décrire les cas de tests à l'avance. On peut se lancer dans la validation d'une fonctionnalité dans un mode appelé le test exploratoire qui consiste à naviguer dans l'application sans plan bien défini.

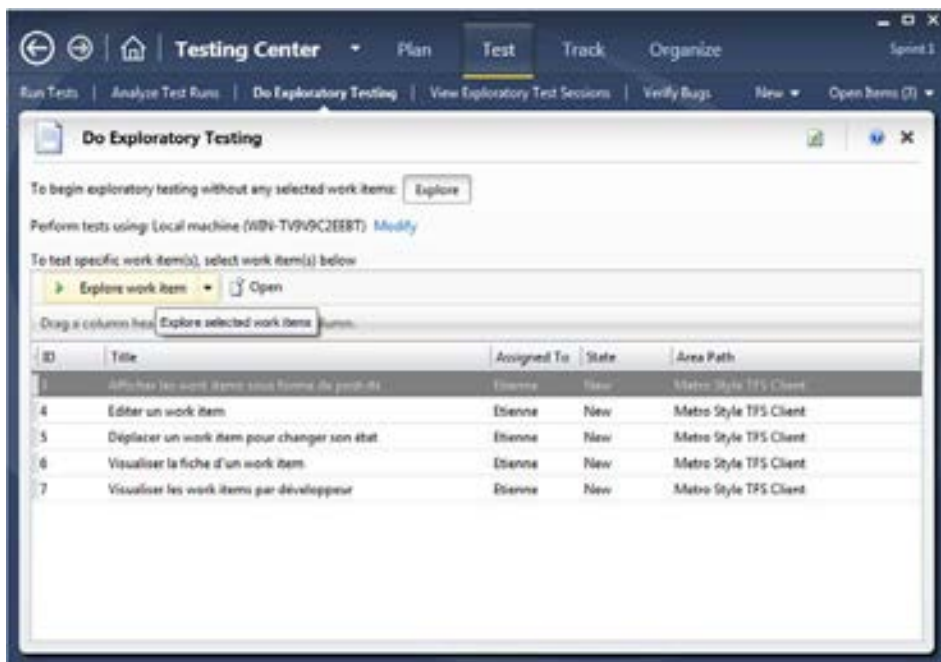


Figure 45 : Démarrer une session de tests exploratoires

Lorsque l'outil est démarré, il est possible de choisir un ensemble de collecteurs, de la même manière qu'on le ferait pour un test classique pour que tout ce qu'on réalise soit enregistré. Au fil de l'eau, le testeur ajoute des commentaires, prend des captures d'écran si nécessaire et enrichit ainsi son exploration.

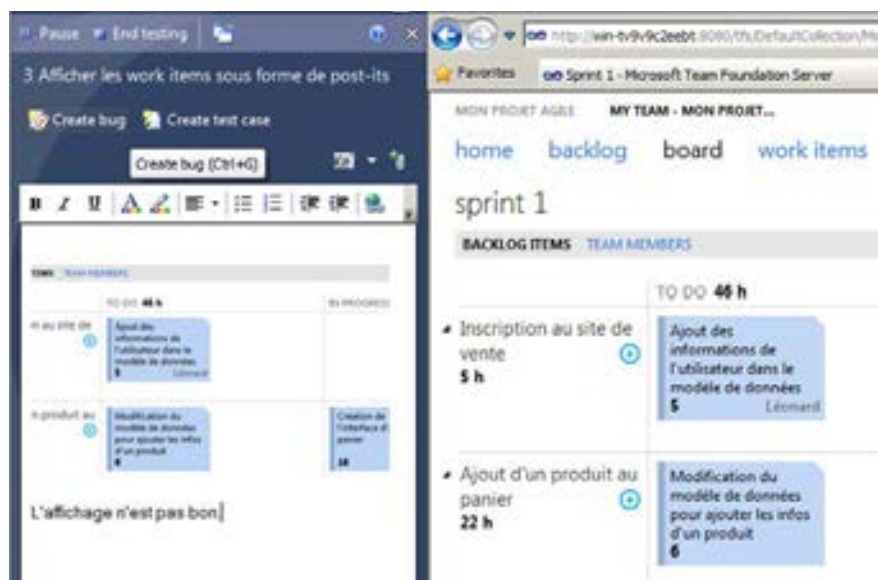


Figure 46 : Créer un bug après un test

Grâce à toutes les informations saisies au fur et à mesure par le testeur et aux informations collectées telles que la vidéo et les actions réalisées au clavier et à la souris, l'opération de création d'une fiche de bug est pré-remplie avec les informations nécessaires à l'analyse du problème.

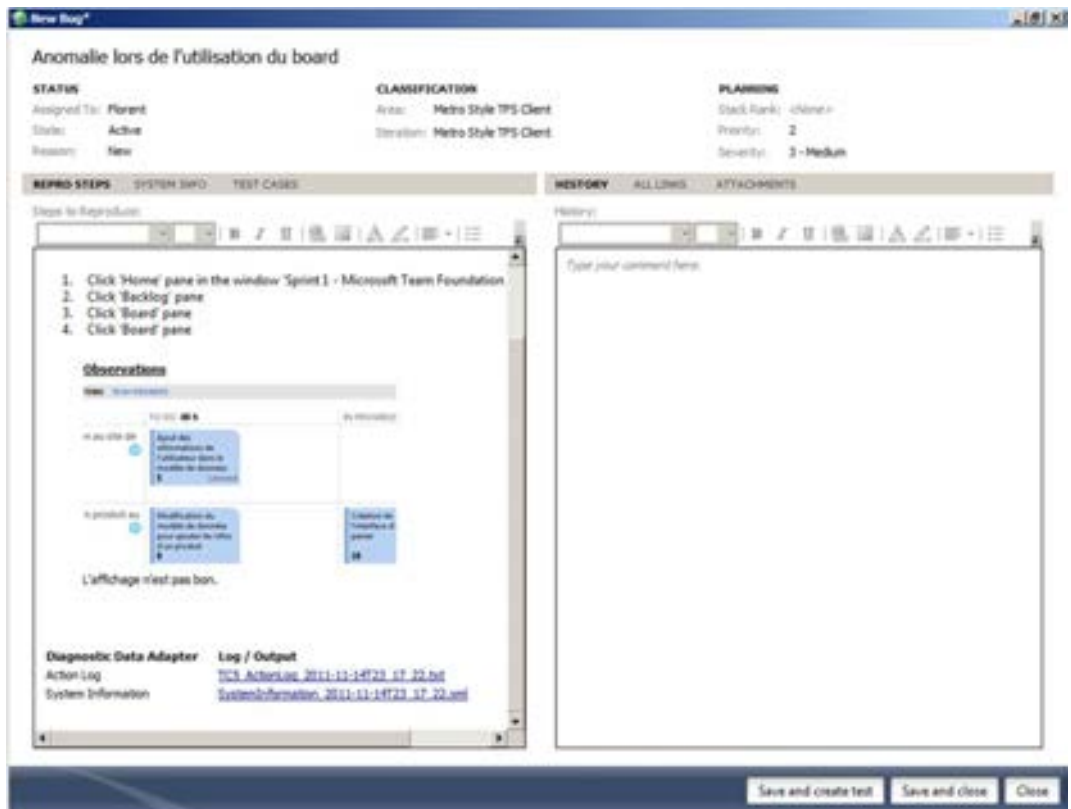


Figure 47 : Fiche de bug déjà remplie par l'outil

Pour plus de pérennité, il est également possible de créer un cas de test à partir d'un test exploratoire. Ceci permet de conserver le travail effectué pour pouvoir le rejouer manuellement par la suite.

Il est important de noter qu'il est possible de capitaliser facilement sur le travail réalisé ici. On peut en effet transformer ce test dans une version automatisée pour pouvoir l'intégrer à une batterie de tests de non régression, pour ne pas perdre de temps lors des prochaines modifications.

## 10. Le besoin de feedback

### A. Feedback produit

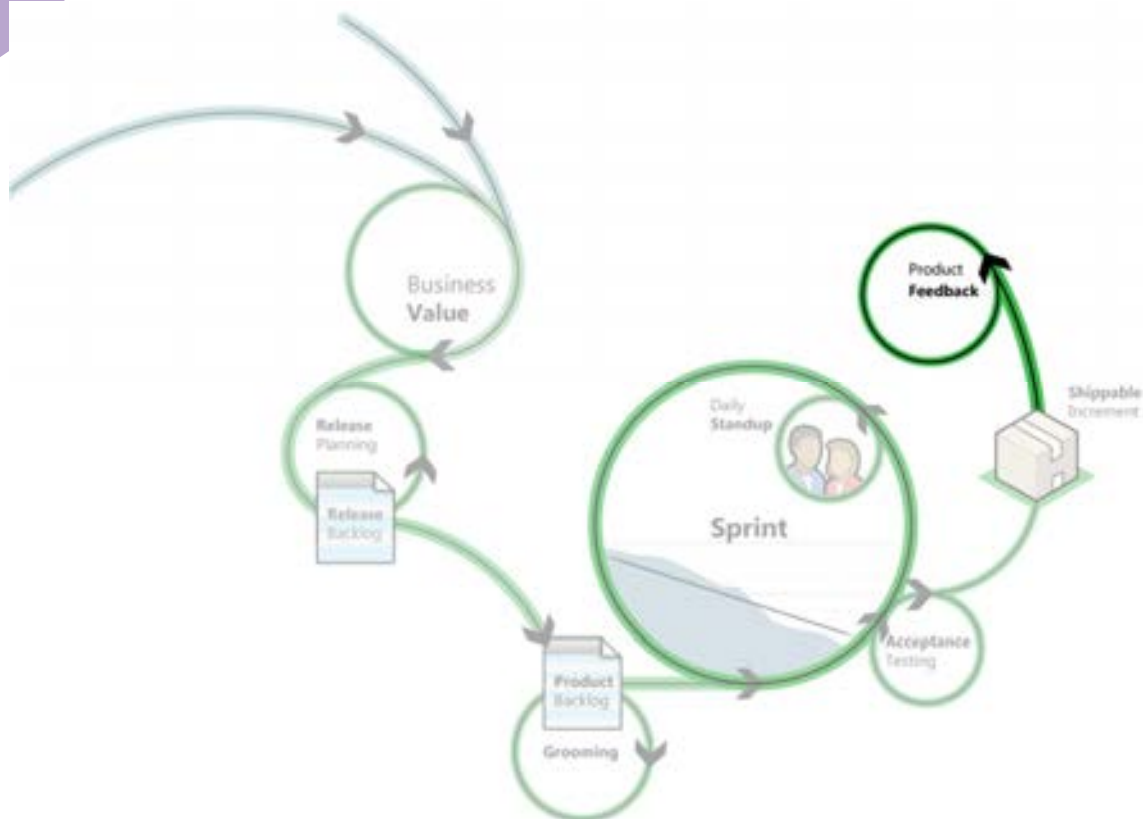


Figure 48 : Scrum, feedback produit

Lors de la revue de sprint, le moment où le Product Owner démontre les différentes stories, il est important qu'il récolte le plus de retours possibles. Ceci permet de s'assurer d'avoir une version correspondant toujours au maximum à ce dont l'utilisateur a besoin.

Lors de cette réunion, le Product Owner peut s'appuyer sur TFS pour obtenir la liste des stories à démontrer et autour desquelles il faut discuter. Il peut également profiter de ce moment pour ajouter de nouvelles stories, ou en supprimer.

Si un feedback intéressant est proposé pendant la réunion, il est possible d'ajouter un élément de travail de type **Feedback** et de l'associer à l'élément du backlog correspondant.

## B. Feedback client

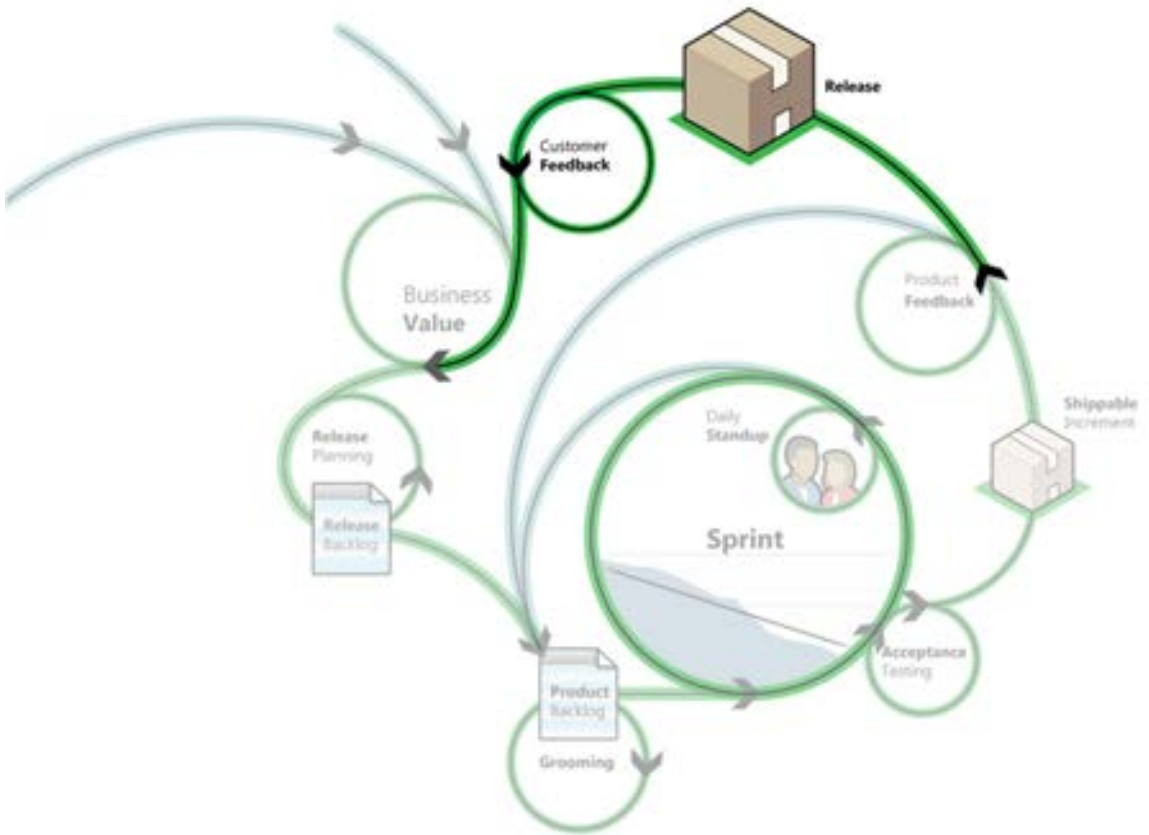


Figure 49 : Scrum, feedback client

Le terme « agile », dans la méthode de développement du même nom, décrit une certaine souplesse et une réactivité plus élevée que dans une méthode plus classique. Être réactif, cela nécessite avoir la possibilité de comprendre ce que l'utilisateur final de l'application souhaite. Cela passe évidemment par le backlog et le détail des différents scénarios d'utilisation, mais également par les retours que font les utilisateurs clés lors de la création d'une fonctionnalité via le **feedback**.

Lorsque l'on termine d'implémenter un cas d'utilisation (ou tout du moins à la fin de l'itération dans laquelle il est inclus), on ne considère qu'il correspond au besoin de l'utilisateur que lorsque celui-ci a pu voir et tester ce qui a été réalisé. Pour faciliter la communication autour d'un feedback d'un utilisateur, la gamme de produit Visual Studio 2012 inclut un nouvel outil : **Microsoft Feedback Manager**. Il s'agit d'un client riche à destination des utilis-

teurs à qui on offre la possibilité de remonter des informations à l'équipe de développement. Ces informations ne sont pas nécessairement des anomalies : il peut s'agir de remarques générales sur le comportement réel et celui attendu.

Le processus classique est pris en compte très simplement dans l'outil : un membre de l'équipe souhaite obtenir du feedback de la part d'un utilisateur. Pour cela, il utilise le site d'équipe, choisit l'élément du backlog pour lequel il souhaite avoir un retour, et envoie automatiquement un mail de demande de feedback.

NEW FEEDBACK REQUEST

To: Etienne ✕  
Display Name or Domain\Username

browse | check name

Subject: Que penses-tu de la réalisation ?

Items for feedback:  
✕ [2] En tant que nerd je veux héberger un dîner

⊕ Add item(s) for feedback

Getting started information

Preview Cancel

Figure 50 : Faire une demande de feedback

Lorsque la demande est effectuée, le destinataire obtient un mail qui lui permet de cliquer sur un lien ouvrant l'outil de feedback. La liste des cas d'utilisation à valider lui est présentée et il lui suffit alors d'utiliser la fonctionnalité associée et d'indiquer au fur et à mesure ses impressions. Tout comme dans le cadre de l'exécution d'un test avec Test Manager, des informations sont enregistrées au fil de l'eau de manière à enrichir le feedback qui sera remonté. Lorsque l'essai est fini, l'utilisateur peut choisir de communiquer le feedback à la personne l'ayant demandé d'un simple clic sur un bouton.

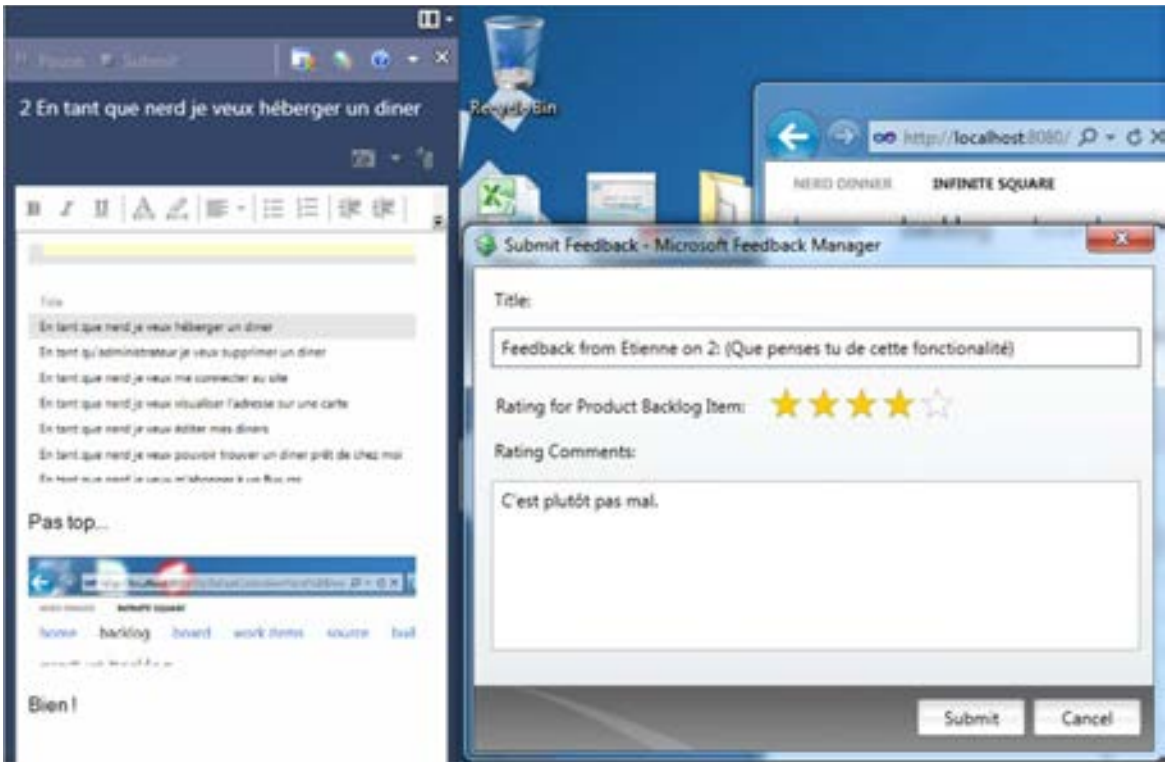


Figure 51 : Outil de feedback

## 11. Suivre l'avancement d'un sprint

La réussite d'un projet dépend en grand partie de la capacité de l'équipe à tracer, suivre et comprendre l'information. Ceci se traduit à un niveau détaillé par un accès aux différents éléments de travail tels que les tâches et les bogues ou à l'historique d'un fichier dans le contrôle de code source. Ce niveau de détail devient trop fourni lorsque l'on essaye d'obtenir une vue globale du projet.

L'avantage qu'apporte TFS par rapport à d'autres solutions est que tout est intégré. La plateforme permet de tout gérer si on le souhaite, du post-it au code. Grâce à ceci TFS possède toutes les données nécessaires à la génération de rapport de suivi !

Toutes les données de TFS sont stockées dans une base de données SQL Server. La base opérationnelle est celle utilisée au quotidien lorsque l'on archive du code ou que l'on ajoute un élément de travail. Pour des raisons évidentes de performances, les rapports ne sont pas générés directement sur cette base. C'est pour cela qu'elle alimente un entrepôt dé-normalisé.



lisé (Warehouse) au sein duquel les données sont organisées spécialement pour pouvoir obtenir les indicateurs croisés et agrégés comme on le souhaite. Enfin, cette base alimente un cube OLAP Analysis Services. Celui-ci est optimisé pour la lecture de données croisées, car tout est calculé et généré lors de l'insertion de l'information. C'est en général les informations du cube qui sont utilisées pour les rapports.

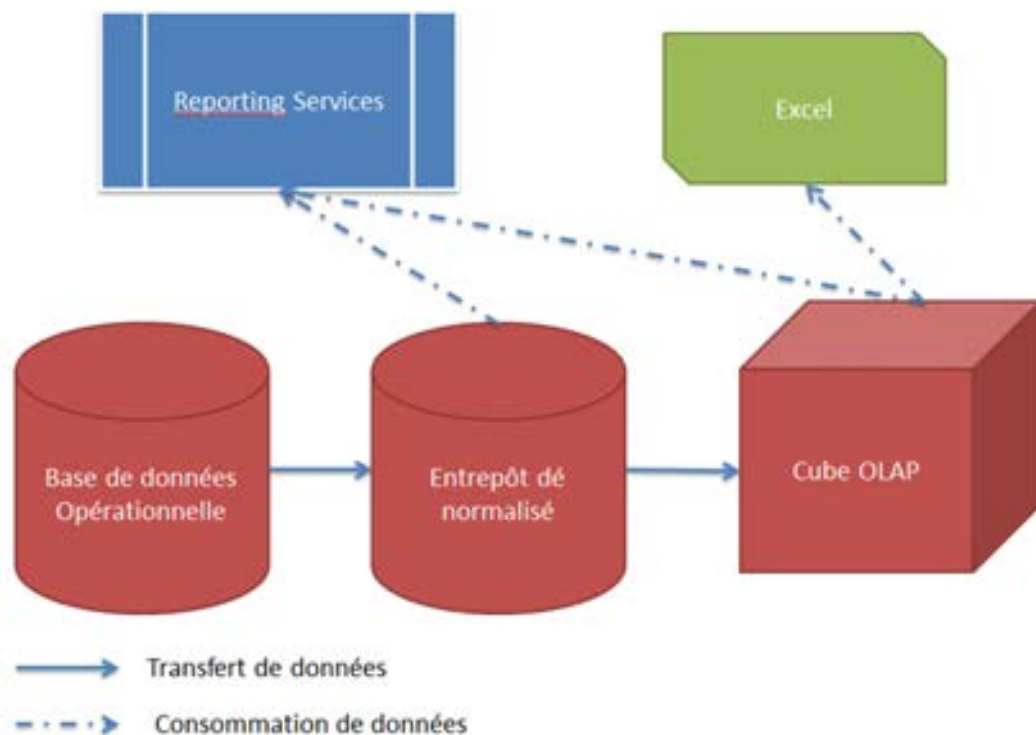


Figure 52 : Les échange de données pour le reporting dans TFS

Avec TFS, les données sont exploitées grâce aux deux outils Reporting Services et Excel.

Reporting services est outil inclus dans la gamme SQL Server. Il s'agit d'un site web au sein duquel il est possible d'ajouter des modèles de rapports qui s'appuient sur une source de données comme par exemple un cube. Les méthodes livrées avec TFS fournissent toujours un ensemble de rapports adaptés à la méthode. Les modèles de processus agiles en ont chacun une liste plus ou moins fournie. Par exemple : le rapport de Burndown permet de visualiser l'avancement du sprint courant, ou de la release en fonction de ce que l'on choisit comme date de début et de fin.



## Release Burndown

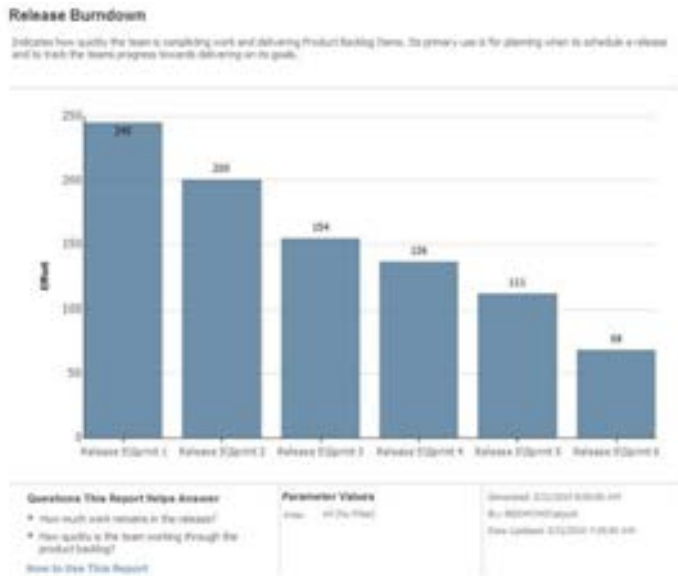


Figure 53 : Release burndown

## Vélocité

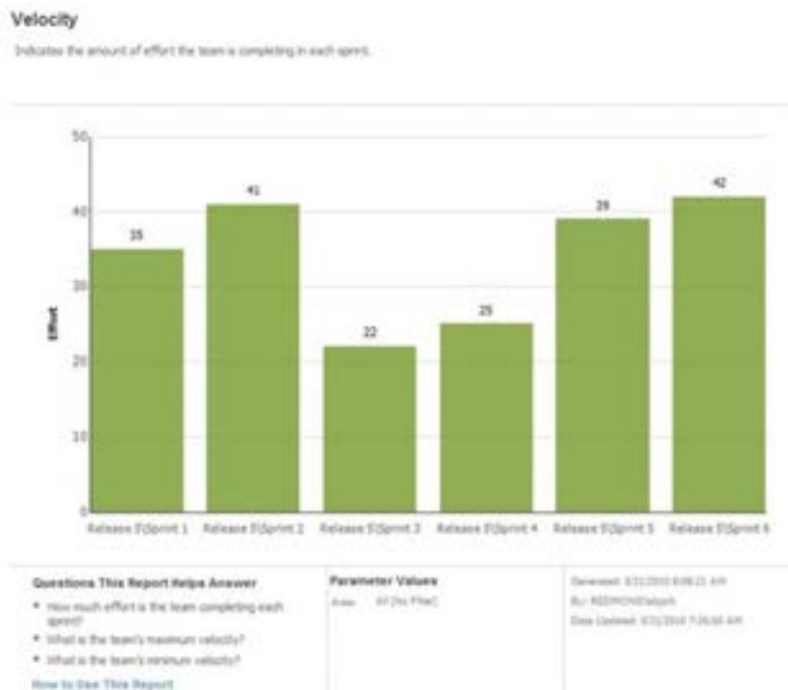


Figure 54 : Vélocité

## Burndown de sprint

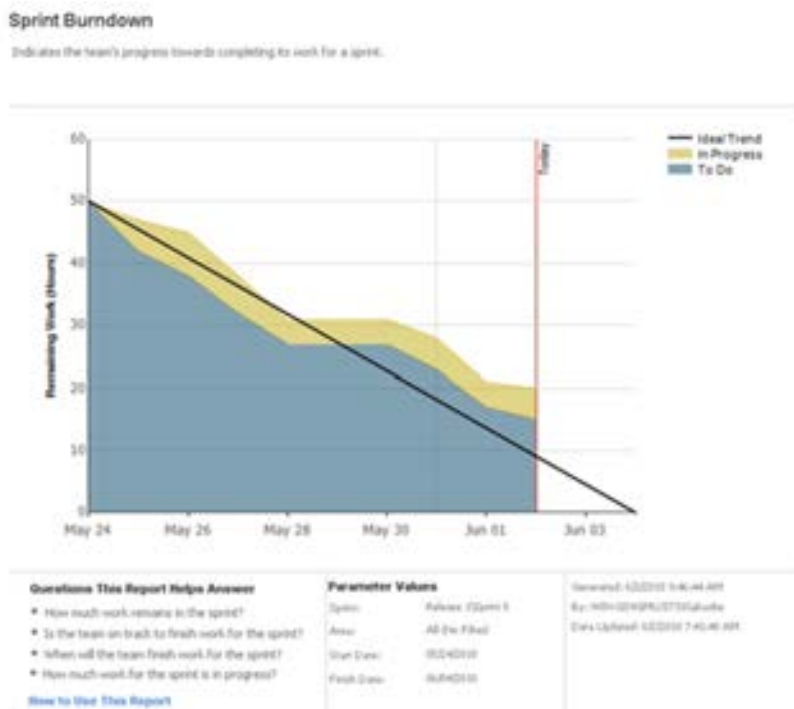


Figure 55 : Sprint burndown

Excel est un outil très largement utilisé pour pouvoir se connecter à une source de données telle qu'un cube Analysis Services pour pouvoir visualiser des données sous forme de tableau croisés dynamiques ou de graphiques synthétiques. Depuis la version 2010 de TFS, le plugin Team Explorer d'Excel contient une fonctionnalité supplémentaire permettant de générer automatiquement un ensemble de rapports à partir d'une vue sur des éléments de travail. L'avantage de cette solution est qu'il est très simple de créer une requête filtrant sur un ensemble de tâches, d'éléments du backlog, de bogues, ou de tout ensemble, par rapport à une itération par exemple. A partir des colonnes que la requête affiche, l'outil est capable de traduire cette vue en requête dans le cube.

Chaque projet TFS est potentiellement associé à un portail SharePoint permettant de partager la documentation. SharePoint, en plus d'être un outil web avancé de gestion de documents apporte la possibilité de construire des pages facilement à partir de morceaux de pages pré-crées et qu'il ne faut que configurer. La version complète de SharePoint possède une fonctionnalité appelée **Excel Services**. Ceci permet d'ajouter un graphique Excel au sein de SharePoint par l'intermédiaire d'un morceau de page spécifique. Il est alors possible de construire très facilement un tableau de bord général constitué d'un ensemble de rapports intéressants.

## Etre agile dans les nuages

---

Depuis plusieurs versions de TFS déjà, des sociétés tierces proposent des solutions de Team Foundation Server hébergées, dans des Data Center ou dans le Cloud. Déjà disponible en beta test, la solution Microsoft de TFS hébergée dans Windows Azure s'appelle Team Foundation Service.

Bénéficiant déjà des nouvelles fonctionnalités du nouveau Team Foundation server 2012, cette version hébergée permet d'utiliser les principales fonctionnalités attendues :

- Gestion du code source et des éléments de travail
- Contrôle de l'intégration continue
- Vérification des tests
- ...

Bref, le plus important y est inclus. Malgré l'absence de la plateforme collaborative SharePoint et de Reporting Services, le rapport de Burndown sera tout de même présent, directement accessible depuis la nouvelle interface de l'accès web. Ceci est largement suffisant pour l'instant, compte tenu du type d'équipe ciblé par la plateforme.

Le plus gros avantage de cette structure est la possibilité d'accéder à une version assez complète de TFS tout en faisant d'abstraction de tous les problèmes liés à l'administration et à la maintenance d'un serveur classique.

Cette solution est tout à fait pertinente pour des entreprises de petite taille, pour qui Team Foundation Service sera un outil parfait pour débiter avec TFS, tout en ne se privant pas de la possibilité de migrer plus tard vers la solution complète Team Foundation Server 2012 hébergée directement par l'entreprise.

Enfin, cette offre ne se limite pas forcément aux petites structures. La scalabilité de la Plateforme Windows Azure permet de supporter un nombre important d'utilisateurs sans pour autant requérir l'achat d'un gros serveur. Cela permet de simplifier les aspects de gestion de load balancing, très loin des préoccupations de la gestion de projet...

## Conclusion

---

Dans le monde du développement logiciel, la popularité des méthodes « Agiles » va croissant. Ces approches partagent les mêmes objectifs : améliorer la visibilité au sein de l'équipe de projet et l'interactivité entre les membres de l'équipe tout en augmentant le flux de valeur pour le client.

Beaucoup de bonnes pratiques sont associées aux méthodes Agiles. Ces pratiques peuvent être adoptées progressivement et appliquées à presque n'importe quel projet ou type de processus, y compris lorsqu'il s'agit d'approche formelle telle que CMMI ou traditionnelle de type « Waterfall ». Comme nous l'avons vu dans ce livre blanc, ces approches proposent des notions comme le backlog, les « itérations » ou « sprints », le daily stand-up, les tableaux de bord, le rapport de burndown, les feedbacks, les tests unitaires, la programmation en binôme et bien d'autres.

Les solutions de gestion du cycle de vie des applications (ALM) doivent permettre de tirer parti des meilleures pratiques Agiles, tout en adaptant et supportant la démarche déjà mise en oeuvre afin de permettre aux équipes de développement d'exploiter progressivement ces pratiques à leur rythme. Elles doivent ainsi fournir une plateforme qui répond aux besoins de l'organisation et à l'ensemble des processus de travail qui y sont définis.

En réponse à ces problématiques, Microsoft propose Visual Studio, une solution intégrée qui brise les silos en favorisant la collaboration entre les différents membres d'une équipe projet tout en assurant une qualité logicielle de bout en bout. Visual Studio 2012 permet de franchir une autre étape dans la couverture du cycle de vie d'une application. L'agilité y est une composante majeure, les outils n'ont jamais été aussi simples et naturels à utiliser, l'objectif étant de favoriser des interactions riches et productives qui augmentent la productivité des équipes et réduisent les risques.

# A propos du Microsoft Technology Center

Les Microsoft Technology Centers (MTC) sont des environnements uniques d'innovation et de collaboration qui offrent l'accès aux dernières technologies et à l'expertise permettant aux Entreprises, Services Publics et Partenaires de concevoir et de déployer les solutions qui répondent exactement à leurs besoins.

Le Microsoft Technology Center a pour triple mission d'apporter aux solutions envisagées une vision stratégique claire et des capacités de démonstration; de participer à la mise en place rapide d'une solution par un transfert de compétences et l'accompagnement des meilleurs experts; et enfin de minimiser les risques par une validation de concept ou un test de performances des solutions envisagées.

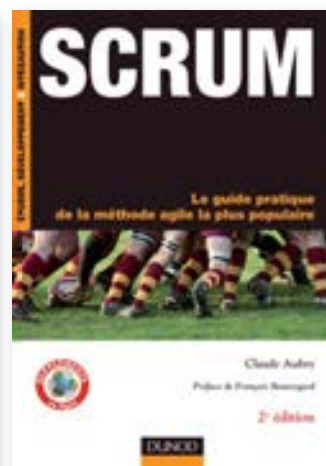
A Paris, le MTC (Microsoft Technology Center) offre une expérience unique d'innovation, et est l'environnement clé, accélérateur de décisions sur les plateformes Microsoft et celles de ses partenaires, matériels et logiciels. Depuis 2004, **architectes du MTC Paris et experts de la plateforme Microsoft bâtissent quotidiennement des solutions et jouent ainsi un rôle essentiel pour relever les défis posés par l'ensemble de ses clients.** Le MTC Paris constitue un environnement de choix pour accélérer les décisions technologiques en proposant un espace de travail de 20 salles collaboratives pour des présentations, démonstrations, ateliers d'architecture et mise en œuvre de prototypes. L'offre ALM est l'une des composantes majeures du catalogue des services proposé par le MTC (<http://blogs.msdn.com/b/mtcparis/archive/tags/alm>)

Pour permettre ces actions, le datacenter du MTC héberge plus de 50 solutions et acteurs concurrents sur une infrastructure dynamique (Private Cloud) de plus de 300 serveurs et 200 To de stockage mis en œuvre avec nos partenaires Alliance. Avec plusieurs centaines de projets par an depuis le démarrage de l'activité, l'équipe MTC apporte un retour d'expérience unique auprès de chaque nouveau projet.

Situé à Paris, au 39 quai du Président Roosevelt, 92130 Issy-les-Moulineaux ; le MTC Paris est l'un des principaux parmi les 20 centres à travers le monde (<http://www.microsoft.com/en-us/mtc/locations/paris.aspx>)

## Annexe : Références techniques

Quelques liens vous permettant d'aller plus loin...



Ressources électroniques :

<http://www.visualstudio.fr>

<http://blogs.developpeur.org/etienne>

<http://blogs.developpeur.org/azra>

<http://blogs.msdn.com/b/mtcparis/>

<http://blogs.msdn.com/b/bharry/>

# SHORTBREAD VISUAL STUDIO à la myrtille

## Ingrédients

### Pour le biscuit :

75g de farine  
25g de sucre  
50g de beurre

### Pour le caramel :

50g de beurre  
50g de vergeoise  
400g de lait concentré  
10cl de purée de myrtille  
2 citrons verts

### Couverture :

200g de couverture ivoire  
colorant violet en poudre

### Croquant pétillant :

100g de couverture ivoire  
20g de sucre pétillant

### Déco : myrtilles fraîches

## Recette

### LE BISCUIT :

Mélanger la farine le sucre et le beurre jusqu'à homogénéité.

Cuire à 180°C pendant 20 minutes. Laissez refroidir.

### LE CARAMEL :

Faire fondre le beurre avec le lait et le sucre en tournant continuellement, lorsque le sucre est dissout porter à ébullition.

Une fois à ébullition réduire la température tout en continuant de tourner et cuire environ 7 minutes ou jusqu'à ce que le caramel ait épaissi. Ajoutez la purée de myrtille et laissez reprendre la consistance désirée. Zestez les citrons verts.

Verser sur le biscuit dans le moule et laisser refroidir.

### LA COUVERTURE :

Une fois refroidi, fondre le chocolat avec le colorant. Versez sur le caramel.

Taper le fond du moule sur le plan de travail pour rapidement répartir le chocolat cela assure une coque en chocolat extra lisse.

### CROQUANT PETILLANT :

Faites fondre le chocolat et coulez le finement sur un silpat. Parsemez de sucre pétillant. Coupez des triangles à l'aide d'un couteau à la lame chaude. Plantez le triangle sur la couverture.

### DRESSAGE :

Découpez le Shortbread Visual Studio en triangle. Piquez de petits triangles blancs de couverture pétillants et parsemez de quelques myrtilles fraîches.



Microsoft





## Les défis Visual Studio



### Relevez les Défis Visual Studio

Montrez ce dont vous êtes capables  
et passez du simple commis au chef étoilé !

[lancez-vous sur visualstudio.fr/defis](http://visualstudio.fr/defis)



Installez l'extension  
Défis Visual Studio  
depuis la galerie Visual  
Studio

**Téléchargez  
l'extension VS**



Accumulez des points  
dans l'environnement  
de développement VS  
2012

**Débloquez les haut  
faits**



Des QR Codes sont  
dissimulés dans tous les  
événements Visual  
Studio. A vous de les  
découvrir !

**Participez aux  
événements**

## Notes





**Infinite Square est une société de conseil, expertise, réalisation, formation, spécialisée sur les technologies Microsoft.**

Créée début 2010, Infinite Square est réputée pour l'expertise technique et la qualité relationnelle de ses consultants, architectes, développeurs, formateurs. La société compte aujourd'hui 25 collaborateurs, disposant tous de multiples certifications Microsoft, parmi lesquels 10 Most Valuable Professionals (MVP), constituant ainsi une densité d'expertise et de savoir-faire sans équivalent sur les technologies Microsoft. Infinite Square est spécialisée sur les technologies de développement d'applications et sur la plateforme applicative Microsoft avec quatre domaines d'intervention : le développement d'applications spécifiques, les solutions collaboratives et de portails avec SharePoint, dans des architectures "on premises" ou "online", les solutions décisionnelles / BI avec SQL Server et la gestion de cycle de vie des applications (ALM) avec Team Foundation Server.

