Master

MAcro-clairances pour les Secteurs TERminaux

Scrum IHM

Annexe I

Frédéric MONJO

Master Pro II Interaction Homme-Machine - 2006 / 2007

ENAC / UPS / UT1

Table de révision

Date	Auteur	Portée	Raison
06/03/2007	Frédéric MONJO	Tout le document	Création du document

Table des matières

Introduction

La méthode présentée ci-après est une adaptation de Scrum classique en y intégrant des pratiques issues des démarches IHM. L'objectif est de tirer parti des nombreux avantages inhérents à la démarche Scrum (adaptabilité, efficacité, direction du projet par le client, réactivité, etc.) tout en comblant une lacune importante : la prise en compte des utilisateurs lors de la conception du logiciel, et plus particulièrement de l'interaction de ceux-ci avec celui-là.

D'un point de vue très général, on peut dire que les approches Agile, si elles sont adaptatives et efficaces, elles sont en revanche trop « centrées client », et ne prennent finalement pas du tout en compte les utilisateurs finaux.

Les approches IHM, quant à elles, placent les utilisateurs au centre de leur démarche, mais adoptent une approche « Big Deisgn UpFront » : toute la conception des interactions est portée au début du projet, avant même que la conception et la réalisation technique soient commencées. Même si cette conception initiale a de bonnes chances d'être de très bonne qualité (grâce à l'intervention permanente des utilisateurs), il y aura très probablement des éléments qui changeront une fois le logiciel final réalisé et mis entre les mains des utilisateurs dans un contexte réel de production. Elles mettent aussi de côté la prise en compte du coût de réalisation des interactions conçues, des choix technologiques limitants du projet, et les différences de compétences entre les corps de métier impliqués (un concepteur IHM sait réaliser des interactions riches qu'un développeur ne sait pas forcément bien faire, et un développeur maîtrise les difficultés liées à la gestion des données présentées dans l'interface, ce qu'un concepteur IHM ne sait pas forcément bien faire).

Cette démarche a été conçue uniquement sur la base des connaissances que j'ai pu acquérir pendant ma formation et mes stages, que ce soit du côté des méthodes Agiles ou du côté IHM. Il n'existe à ce jour que peu d'articles traitant du rapprochement de ces deux mondes, la plupart se limitant plutôt à une analyse comparative des pratiques plus ou moins incompatibles.

Ce document est structuré en deux parties : la première rappelle les principes et pratiques fondamentaux de Scrum, et la deuxième présente les adaptations IHM proposées.

I - Processus Scrum

Pour tout savoir sur le processus Scrum, vous pouvez consulter l'article Scrum de Wikipedia(Fr), rédigé par Frédéric Monjo. La description dans ce document sera plus succincte et posera les bases minimales nécessaires pour bien comprendre son fonctionnement.

1 - Principes fondamentaux des méthodes « Agiles »

Les méthodes « Agiles » ont été réunies sous cette dénomination en 2001, après le rassemblement d'industriels et experts de l'industrie, pour tenter de résoudre des problèmes classiques de l'industrie via une nouvelle approche projet. Cette table ronde a abouti à la rédaction du manifeste Agile, qui pose quatre grands principes fondamentaux :

- Favoriser les **individus et leurs interactions** aux **processus et outils**
- Favoriser du **logiciel qui fonctionne** à une **documentation exhaustive**
- Favoriser la **collaboration du client** à la **négociation de contrat**
- Favoriser la **réponse au changement** au **suivi d'un plan prédéfini**

2 - Idées clé

- Le client est au coeur du projet
- Esprit d'équipe
- La communication est la clé
- Penser simple, efficace, et Qualité
- Flexibilité aux changements
- Avancement basé sur du concret

3 - Rôles dans Scrum

Il n'y a que très peu de rôles dans Scrum, et il n'y a notamment pas de rôles côté MOE. En effet, une des idées fortes est que l'équipe travaille en collaboration permanente, avec une pensée et une réflexion communes. La définition de rôles et de responsabilités n'est alors plus utile, c'est l'équipe entière qui est concernée. Cela n'empêche pas que chaque équipier puisse avoir son domaine de prédilection, mais il n'y est pas borné et il est impliqué dans toutes les phases de la réalisation du produit.

Scrum définit ainsi quatre rôles :

- Le **Directeur de produit** : c'est le représentant des clients et utilisateurs. C'est lui qui définit l'ordre dans lequel les fonctionnalités seront développées, et qui prend les décisions importantes concernant l'orientation du projet. Le terme Directeur n'est d'ailleurs pas à prendre au sens hiérarchique du terme, mais dans le sens de l'orientation.

- Le **Scrum Master**, qui joue un rôle capital : c'est lui qui est chargé de protéger l'équipe de tous les éléments perturbateurs extérieurs à l'équipe et de solutionner ses problèmes non techniques (administratifs par exemple). Il doit aussi veiller à ce que les valeurs de Scrum soient appliquées, mais il n'est pas un chef de projet ni un intermédiaire de communication avec les clients.

– L'**équipe** : elle ne comporte pas de rôles prédéfinis, elle est auto-gérée. Il n'y a pas non plus de notion de hiérarchie interne : toutes les décisions sont prises ensemble, et personne ne donne d'ordre à l'équipe sur sa façon de procéder. Contrairement à ce que l'on pourrait croire, les équipes auto-gérées sont celles qui sont les plus efficaces et qui produisent le meilleur niveau de qualité de façon spontanée. L'équipe s'adresse directement au Directeur de produit. Il est conseillé qu'elle lui montre le plus souvent possible le logiciel développé.

– Les **intervenants** (*Stakeholders*) : ce sont les personnes qui souhaitent avoir une vue sur le projet sans réellement s'investir dedans. Ils peuvent participer aux discussions, mais n'ont aucun pouvoir de décision. Il peut s'agir par exemple d'experts techniques ou d'agents de direction qui souhaitent avoir une vue très éloignée de l'avancement du projet.

4 - Gestion des besoins

a - Backlog de produit

Scrum utilise une approche fonctionnelle pour récolter les besoins des utilisateurs. L'objectif est d'établir une liste de fonctionnalités à réaliser, que l'on appelle backlog de produit.

A chaque item de backlog sont associés deux attributs : une estimation en points arbitraires (voir l'article sur Wikipedia pour les techniques d'estimation), et une valeur client, qui est définie par le Directeur de produit (retour sur investissement par exemple). Ce dernier définit dans quel ordre devront être réalisés ces items. Il peut changer cet ordre en cours de projet, et même ajouter, modifier, ou supprimer des items dans le backlog.

La somme des points des items du backlog de produit constitue le reste à faire total du projet. Cela permet de produire un release burndown chart, qui montre les points restant à réaliser au fur et à mesure des sprints.

b - Backlog de sprint

Lorsqu'on démarre un sprint, on choisit quels items du backlog de produit seront réalisés dans ce sprint. L'équipe décompose ensuite chaque item en liste de tâches élémentaires (techniques ou non), chaque tâche étant estimée en heures et ne devant pas durer plus de 2 jours. On constitue ainsi le backlog de sprint.

Pendant le déroulement du sprint, chaque équipier s'affecte des tâches du backlog de sprint et les réalise. Il met à jour régulièrement dans le backlog du sprint le reste à faire de chaque tâche. Les tâches ne sont pas réparties initialement entre tous les équipiers, elles sont prises au fur et à mesure que les précédentes sont terminées.

5 - Lancement du projet

Scrum présuppose que le backlog de produit est déjà défini au début du projet. Une approche possible pour constituer ce backlog est de réaliser une phase de lancement. L'expression initiale des besoins n'est surtout pas une activité de contractualisation d'un cahier des charges. L'objectif pour Scrum est de produire la première version du backlog de produit.

L'esprit d'une telle activité dans les méthodes Agiles est de définir d'une part le cadre du projet (pour que l'équipe s'imprègne du contexte métier), et d'autre part une première analyse globale des besoins. L'objectif est d'identifier un maximum de

fonctionnalités que le logiciel devra implémenter, en se limitant à un niveau de précision assez grossier. On peut par exemple utiliser une approche par raffinages successifs, en partant des secteurs métier concernés par l'application, puis en identifiant les activités métier, qu'on décompose en tâches métier qui correspondent à des fonctionnalités que l'on doit/peut ou non implémenter dans le logiciel final.

6 - Planification du projet

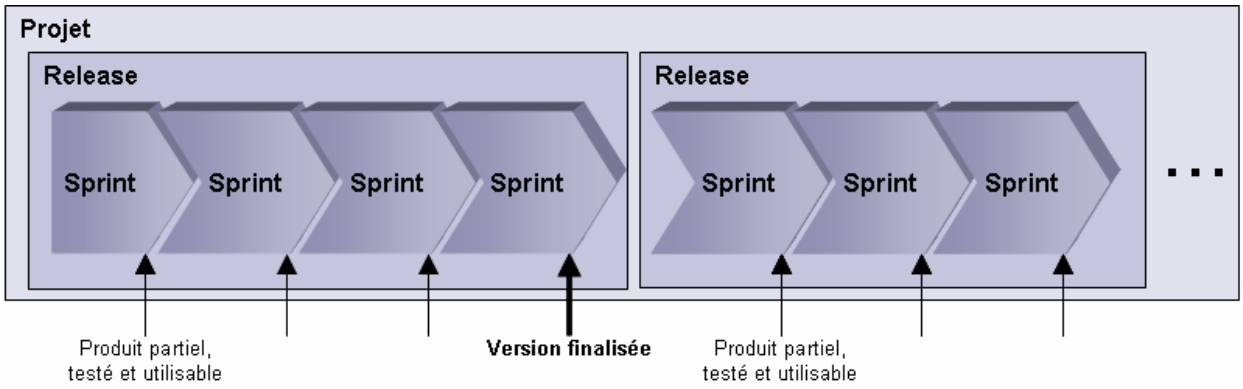

Figure 1 : Planification en Scrum

Scrum est une méthode dite "itérative" : le produit final est construit par incréments successifs. Une itération dure entre 2 et 4 semaines, mais leur durée, définie en début de projet ne doit pas changer d'une itération à l'autre. Chaque incrément donne lieu à une version fonctionnelle intermédiaire du produit. Pendant une itération, il est interdit de modifier quoi que ce soit à ce qui était prévu. En revanche, entre les itérations, il est possible (et probable) de changer l'ordre des fonctionnalités réalisées, d'en ajouter ou d'en enlever, etc.

a - Sprint

Le **sprint** est le nom donné à une itération. Chaque sprint possède un but, et on lui associe une liste d'items de backlog de produit (fonctionnalités) à réaliser. Ces items sont décomposés par l'équipe en tâches élémentaires de quelques heures, les items de backlog de sprint.

Pendant un sprint, la liste des items de backlog de produit à réaliser ne peut pas être changée. Les changements éventuels sont pris en compte dans le backlog de produit, et seront éventuellement réalisés dans les sprints suivants.

Il y a une exception à cela : il se peut que l'équipe se rende compte en cours du sprint qu'elle n'aura pas le temps de terminer un item du backlog de produit, ou au contraire qu'elle aura fini en avance. Dans ce cas, et seulement d'un commun accord entre l'équipe et le directeur du produit, on peut enlever ou ajouter un item à ce qui avait été prévu.

b - Release

Pour améliorer la lisibilité du projet, on regroupe généralement des itérations en **releases**. Ce concept est utilisé pour mieux identifier les versions. En effet, comme chaque sprint doit aboutir à la livraison d'un produit partiel, une release permet de marquer la livraison d'une version aboutie, susceptible d'être mise en exploitation.

Il est intéressant de planifier à l'échelle d'une release en répartissant les items du backlog de produit sur les sprints, tout en respectant leur priorité. Bien entendu, ce qui est planifié au-delà du sprint courant peut changer à tout moment.

c - Quotidien

Au quotidien, une réunion, le **Scrum**, permet à l'équipe et au ScrumMaster de faire un point d'avancement sur les tâches et sur les difficultés rencontrées. Cette réunion dure 15min au maximum.

7 - Déroulement d'un sprint

a - Réunion de planification du sprint

Tout le monde est présent à cette réunion, qui ne doit pas durer plus de 4 heures. La réunion de planification (*Sprint Planning*) consiste à définir d'abord un but pour le sprint, puis à choisir les items de backlog du produit qui seront réalisés dans ce sprint. C'est le directeur de produit qui prend ces décisions, en accord avec l'équipe, qui possède l'expertise technique pour juger de la faisabilité.

Dans un second temps, l'équipe décompose chaque item du backlog de produit en liste de tâches (items du backlog du sprint), puis estime chaque tâche en heures. Il est important que le directeur de produit soit présent dans cette étape. En effet, il est possible que des tâches le concernent (comme la rédaction des règles métier que le logiciel devra respecter et la définition des tests fonctionnels).

b - Au quotidien

L'équipe travaille dans une même pièce, dont le ScrumMaster doit maintenir la qualité de son environnement. Les activités se déroulent éventuellement en parallèle : analyse, conception, codage, intégration, tests, etc. Scrum ne définit volontairement pas de démarche technique pour le développement du logiciel : l'équipe décide en toute autonomie de la façon dont elle va travailler.

L'équipe utilise la démarche du développement guidé par les tests. Cela consiste à coder en premier lieu les modules de test vérifiant les contraintes métier, puis à coder ensuite le logiciel à proprement parler, en exécutant les tests régulièrement. Cela permet de s'assurer entre autres de la non régression du logiciel au fil des sprints.

c - Revue de sprint

A la fin du sprint, tout le monde se réunit pour effectuer la revue de sprint, qui dure au maximum 4 heures. L'objectif de la revue de sprint est de valider le logiciel qui a été produit pendant le sprint. L'équipe commence par énoncer les items du backlog de produit qu'elle a réalisés. Elle effectue ensuite une démonstration du logiciel produit. C'est sur la base de cette démonstration que le directeur de produit valide chaque fonctionnalité planifiée pour ce sprint.

Une fois le bilan du sprint réalisé, l'équipe et le directeur de produit proposent des aménagements sur le backlog du produit et sur la planification provisoire de la release. Il est probable qu'à ce moment des items soient ajoutés, modifiés, ou réestimés, en conséquence de ce qui a été découvert.

d - Rétrospective du sprint

La Rétrospective du sprint est faite en interne par l'équipe (incluant le ScrumMaster). L'objectif est de comprendre ce qui n'a pas bien marché dans le sprint, les erreurs commises, et de prendre des décisions pour s'améliorer. Il est tout à fait possible d'apporter des aménagements à la méthode Scrum dans le but de s'améliorer.

II - Aménagements IHM

La méthode Scrum a été enrichie avec les quelques aménagements présentés ci-après, issus des pratiques IHM.

Ces aménagements, inspirés de quelques publications d'experts et des cours du Master Pro II IHM, ne relèvent d'aucune démarche explicitement reconnue du monde industriel. Ils sont donc testés pendant le déroulement de notre chef d'oeuvre, et peuvent ou non être appliqués au cas par cas.

1 - Phase de lancement

Comme mentionné plus haut dans le descriptif de Scrum, il n'y a pas de démarche explicitement définie dans cette méthode pour aboutir à un backlog de produit. La démarche ci-dessous utilise quelques techniques qui permettent d'avoir une première version du backlog assez étoffée. Cette phase de lancement se déroule suivant les étapes qui suivent.

Nous verrons qu'elle révèlera plutôt des fonctionnalités métier, chose primordiale, mais l'équipe devra probablement intégrer des item de nature plus technique (contraintes, etc.) issus de son expertise.

a - Compréhension générale du domaine métier

Cette étape consiste à donner une compréhension globale du domaine du problème. Il ne s'agit pas d'aller dans un niveau de détail trop élevé, mais au contraire de donner une vision générale qui permet de comprendre la place du système à produire dans le contexte métier de l'entreprise. Une bonne durée pour cette phase est de 2 heures maximum.

b - Préconception du système

Cette étape vise à recueillir un maximum d'idées de ce que pourrait faire le système, sans cohérence ou pertinence particulières. C'est plutôt un brainstorming autour de la question : *« Si le système était magique, que ferait-il ? »*

c - Rôles utilisateurs et interactions

Cette étape est un brainstorming dont le but est d'identifier un maximum de rôles métier, qui pourront devenir des rôles utilisateurs du système, puis de définir les interactions entre ces rôles. Elle peut se dérouler de cette façon :

- Brainstorming : trouver tous les rôles impliqués.
- Raffinage de chaque rôle :
  - Formalisme de nomenclature : « *Faiseur de quelque chose [détails]* ».
  - Buts/intentions de son métier.
  - Profil psychologique.
  - Connaissance du processus métier de l'entreprise.
  - Connaissance des technologies informatiques.
- Définition des interactions entre les rôles :

- Faire un diagramme qui montre les interactions entre les rôles. Ces interactions ne sont pas celles du futur système, mais bien celles du métier.
- Identifier quelles interactions pourraient être assistées/automatisées par le système.

d - Tâches utilisateurs

En partant des rôles identifiés dans l'étape précédente, il s'agit de déterminer les tâches de chaque rôle. Ces tâches sont relativement liées aux buts du rôle, mais un même but peut engendrer plusieurs tâches.

Pour exprimer une tâche, on peut utiliser le formalisme simple : *« Faire quelque chose [dans tel but] »*. Attention, il ne faut pas mentionner de quelle façon cette tâche est réalisée (cette façon sera probablement modifiée par le système à réaliser).

e - Créer les items de backlog du produit

La liste des tâches obtenue à l'étape précédente est un bon point de départ pour générer le backlog de produit initial. En effet, pour mémoire, chaque item correspond à une fonctionnalité que le système devra réaliser. Bien qu'une tâche ne corresponde pas forcément exactement à une fonctionnalité, elle est un excellent point de départ pour identifier les fonctionnalités qui découlent de l'accomplissement de ces tâches.

Un bon formalisme pour exprimer les items du backlog de produit est celui des *User Stories* (méthode eXtreme Programming) : « *En tant que* ***Rôle****, je (veux/peux)* ***Faire quelque chose*** *[dans tel but] »*. Le « faire quelque chose » correspond bien sûr à une fonctionnalité du système.

f - Organiser le projet

Cette étape vise à préparer la planification globale du projet. Dans un premier temps, l'équipe estime chaque item du backlog de produit avec un nombre de points, puis propose une vélocité initiale estimée (nombre de points réalisés dans une itération).

De son côté, le Directeur de produit définit l'ordre dans lequel ces fonctionnalités seront réalisées, selon ses propres critères (métier, ROI, etc.). Le client, sur les conseils de l'équipe, devrait bien entendu baser son approche sur les risques, de façon à éliminer les plus importants dans les premières itérations. A ce sujet, il peut être intéressant d'associer à chaque item du backlog les critères usuels d'évaluation des risques (probabilité, impact).

Les items sont ensuite répartis sur l'ensemble des sprints du projet, ce qui permet d'avoir une idée approximative des échéances clés du projet.

g - Conception initiale légère

Il est indispensable, avant de démarrer la réalisation du système, d'avoir une vue globale minimum sur ce dernier. Cette vue globale concerne plusieurs aspects :

- Technologies utilisées : si c'est possible, il faut choisir les technologies utilisées pour le projet, en considérant bien entendu les technologies du système d'information existant de l'entreprise.
- Architecture globale : une première conception de l'architecture du système est incontournable. Cette architecture ne doit pas définir l'intégralité du système, mais plutôt une structure fondamentale robuste et surtout **flexible** et **évolutive**. Un bon

niveau de détail serait celui des design patterns mis en jeu dans l'architecture, sans rentrer dans le détail de leur implémentation.

– Si l'architecture s'annonce complexe, ou qu'un risque technologique est à craindre, il est plus pertinent de dédier les premières itérations à leur mise au point globale.

h - Conclusion

Ce qu'il faut retenir de cette phase de lancement du projet, c'est qu'il est indispensable d'avoir une vision claire du projet, sous deux angles :

– Besoins : la liste des besoins doit être la plus complète possible, même si elle sera probablement amenée à changer au fil des itérations.

– Architecture et technologie : il est important d'être clair sur leurs aspects fondamentaux, mais il n'est pas question ici d'aller dans le détail. L'esprit à avoir est de définir des bases solides, souples et évolutives, les détails étant affinés lors de l'implémentation pendant les sprints successifs.

2 - Aménagements sur une Release

Une release est un ensemble de sprints aboutissant à une version de logiciel stable qui sera déployée.

a - Début de release

Une release est un sous-ensemble du produit final, qui se suffit à lui-même pour apporter immédiatement de la valeur au client. Il est par conséquent intéressant d'avoir une vue macroscopique un peu plus détaillée que celle du début du projet.

On peut ainsi en début de release faire quelques **séances de brainstorming de conception participative** avec les clients, les utilisateurs et l'équipe. L'objectif n'est pas d'obtenir une vue précise sur l'ensemble de l'interface et des interactions, mais plutôt d'identifier des idées clés qui serviront de base lors des raffinages successifs pendant les sprints.

b - Fin de release

A la fin d'une release, il est courant que le dernier sprint soit consacré entièrement à la mise en production de cette release. Voici comment pourrait se dérouler le dernier sprint d'une release :

– **Evaluation du produit** : une release du produit est destinée à la production, aussi il est indispensable qu'elle soit optimale au niveau de son utilisabilité. On peut donc faire une évaluation d'utilisabilité complète, puisqu'on dispose de temps suffisant pour la mener. Au terme de cette évaluation, des **corrections et améliorations** auront peut-être été identifiées : le Directeur de produit et l'équipe peuvent alors décider ensemble de les réaliser immédiatement, ou de les différer dans la release suivante (en les intégrant dans le backlog de produit), en fonction du travail nécessaire et du temps disponible dans le sprint.

– **Déploiement** : une fois les corrections éventuelles effectuées, l'équipe déploie le logiciel sur les postes des utilisateurs finaux. C'est une activité qui peut nécessiter beaucoup de temps suivant la taille du système.

– **Formation des utilisateurs** : bien que les manuels utilisateur aient été rédigés à chaque sprint pour les fonctionnalités réalisées, une formation des utilisateurs est généralement préférable pour réduire leur temps de prise en main du produit.

3 - Aménagements sur un Sprint

A l'échelle d'un sprint, nous avons ajouté des étapes pendant sa planification, et à la revue du sprint.

a - Planification du sprint

Les utilisateurs expriment des besoins d'une certaine façon, mais il y a toujours un décalage entre ce qu'ils disent et ce qu'ils voulaient vraiment, entraînant des changements de besoins. Scrum prévoit complètement ces changements, mais ils ne sont généralement identifiés qu'après la revue du sprint, une fois le logiciel partiel livré. Pour pallier à ce problème, l'idée est d'utiliser les techniques IHM pour anticiper ces changements.

Pour cela, la phase de planification du sprint est décomposée comme suit (les étapes entre crochets sont facultatives, à réaliser seulement dans le cas de systèmes critiques ou très complexes) :

– **Choix des items du backlog de produit à réaliser** (pas de changement par rapport à Scrum classique). C'est la seule étape qui se fait avec la présence des clients.

– **[Interviews et observation]** : il s'agit d'interviewer les utilisateurs sur leur façon actuelle de réaliser les tâches (fonctionnalités) que l'équipe va devoir implémenter. Ces interviews filmées peuvent utiliser entre autres la technique de « l'incident critique ». L'aboutissement de ces interviews peut éventuellement être un ensemble de **Scénarios de travail**. Comme tout document dans Scrum, ces scénarios de travail ne doivent être produits que si leur nécessité est évidente.

– **[Analyse des tâches]** : il est possible d'effectuer une analyse des tâches de l'utilisateur, à l'aide de modèles adaptés pour cela. Cette activité est coûteuse en temps, aussi il n'est intéressant de la mener de façon approfondie que si l'activité des utilisateurs est vraiment complexe.

– **Brainstorming oral de conception** : le but est d'énumérer un maximum d'idées concernant les interactions liées aux fonctionnalités à implémenter dans le sprint. A la fin de la séance, on choisit les meilleures idées, et on les transcrit en maquettes papier. Ces solutions serviront de tremplin pour alimenter les séances de conception participative qui suivent. Elles pourront être utilisées pendant des « creux » pour faire naître de nouvelles idées.

– **Séance(s) de conception participative** : en partant des solutions préliminaires de l'étape précédente, on réalise un brainstorming avec les utilisateurs finaux pour concevoir l'interface et l'interaction du système. Pour cela, on utilise des comme support des prototypes papiers. A la fin de la séance, on choisit les meilleures idées, puis on filme des séquences vidéo des maquettes papier animées pour chaque tâche/fonctionnalité, en expliquant les actions réalisées et le comportement de l'interface.

– **Design walkthrough** : les concepteurs discutent « en interne » les solutions trouvées à l'étape précédente. L'objectif est d'identifier un maximum de problèmes (constructifs). Ces observations servent ensuite à déterminer les maquettes qui seront conservées et développées. Les maquettes papier sont affinées (mises au propre). On peut également mettre au point des **Scénarios de conception**, si cela semble nécessaire : ces scénarios serviront de base de travail pour la réalisation finale des fonctionnalités/tâches concernées. Ils peuvent aussi être utilisés pour la validation des fonctionnalités par le client, à la fin du sprint.

Les résultats de ces premières étapes sont des éléments qui serviront de référence pour la réalisation du système : **vidéos d'interviews et dépouillage, vidéos de prototypes papier, modèles de tâches utilisateurs, scénarios de conception**.

On retrouve ensuite les étapes classiques de Scrum :

– **Conception détaillée** du noyau fonctionnel, produisant si besoin : diagramme de classes, statecharts, etc.

– **Décomposition des tâches du sprint et estimation**

b - Revue du sprint

La revue du sprint est enrichie de la façon suivante :

– **Démonstration du produit** : l'équipe fait une démonstration du logiciel produit aux clients, de façon classique en Scrum, mais *sans les utilisateurs finaux*. Ceux-ci interviennent à l'étape suivante.

– **Evaluation d'utilisabilité** : l'objectif est de faire une évaluation "allégée" du logiciel produit. La préparation de ces tests est une étape *cruciale*, qui aura dû être faite pendant le sprint, au même titre que le reste de la production du logiciel. Pour ne pas consommer trop de temps, on peut évaluer deux utilisateurs en même temps pour chaque rôle : de cette façon, ils discutent spontanément de ce qu'ils font et peuvent émettre des idées nouvelles et inattendues. On pourra utiliser une démarche d'évaluation traditionnelle : introduction de la séance et des tâches que l'utilisateur doit réaliser, explication des principaux concepts de l'interface, déroulement des tests, debriefing des testeurs.

– **Bilan du sprint** : étape Scrum classique.

Conclusion

L'application pure et simple de cette démarche ne saurait garantir que les interfaces et interactions produites soient de bonne qualité. En effet, impliquer les utilisateurs permet d'éviter des aberrations qui peuvent émaner de développeurs non sensibilisés aux IHM, mais cela ne garantit en rien que les interactions seront optimales. Cela nécessite en effet les compétences d'un concepteur IHM qualifié, issues de domaines tels que l'ergonomie, les facteurs humains, la psychologie, la sociologie, l'infographie, etc.

Aussi, pour rejoindre l'esprit des méthodes Agiles qui préconisent et favorisent une mutualisation des compétences dans l'équipe, il serait judicieux d'inclure un spécialiste de la conception d'IHM dans toute équipe, qui participerait au déroulement du projet aux mêmes titre et rang que tous les autres équipiers.

Cette démarche expérimentale est une première tentative pour réunir les mondes IHM et Agile, dans le but de tirer profit des avantages des deux. Elle est probablement sujette à polémique, que ce soit dans un monde comme dans l'autre ; j'espère que ce débat aura lieu et permettra de faire évoluer nos pratiques projet dans un sens constructif.

Schéma récapitulatif global

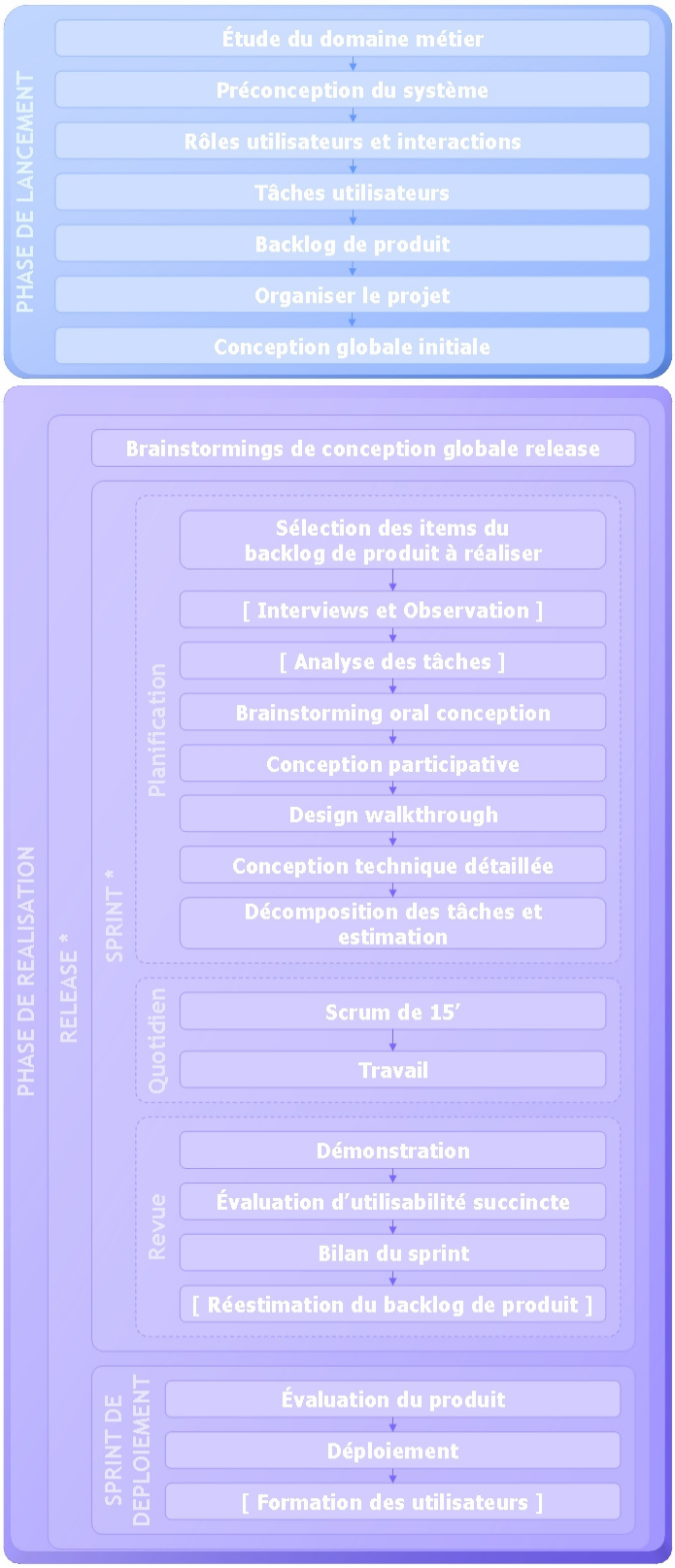

Figure 2 : Schéma récapitulatif global du processus Scrum IHM

Table des illustrations

Glossaire et acronymes

Directeur de produit (Product Owner)	Personne responsable de produire et maintenir à jour le **backlog** de produit. C'est lui qui en détermine les priorités et qui prend les décisions concernant l'orientation du projet.
ScrumMaster	Membre de l'équipe dont l'objectif principal est de la protéger des perturbations extérieures. Il est complètement transparent pour la communication entre l'équipe et les clients, et n'a aucun pouvoir hiérarchique sur l'équipe. C'est en revanche un facilitateur pour les problèmes non techniques de l'équipe.
Backlog de produit (Product Backlog)	Liste des fonctionnalités qui devront être réalisées par le logiciel.
Backlog de sprint (Sprint Backlog)	Liste des tâches à accomplir pendant un sprint. Elles correspondent à la réalisation des items de backlog du produit affectés au sprint.
Scrum	Réunion quotidienne de 15 minutes, qui a pour but de faire le point sur ce qui a été fait depuis le dernier Scrum, ce qui est prévu de faire jusqu'au prochain et quelles sont les embûches rencontrées durant le travail.
Sprint	Nom d'une itération dans Scrum. Cette itération dure 30 jours calendaires en théorie, mais en pratique on utilise plutôt entre 2 et 4 semaines. Pendant une itération, l'équipe doit développer une liste d'items du backlog de produit qui a été définie au début de ce sprint.
Burndown Chart	Graphique qui montre la tendance du reste à faire total de jour en jour (pour les sprints) ou de sprint en sprint (pour les releases).
Brainstorming	« Le remue-méninges, ou brainstorming (littéralement : tempête dans le cerveau), est une méthode de travail en groupe pour obtenir un nombre important d'idées. » *Extrait de la définition sur Wikipedia{fr}*
Conception participative	« La conception participative est une méthode de travail utilisée principalement en conception de logiciel interactif. Sa principale caractéristique est la participation active des utilisateurs au travail de conception. Il s'agit donc d'une méthode de conception centrée sur l'utilisateur où l'accent est mis sur le rôle actif des utilisateurs. » *Extrait de la définition sur Wikipedia{fr}*
Scenario de travail	« Un scénario de travail est la description d'une séquence

	d'événements qui illustre les activités d'une ou plusieurs personnes engagées dans une tâche. » [1]
Scenario de conception	« Un scénario de conception est essentiellement une version améliorée d'un scénario de travail, auquel on ajoute une proposition de nouveau système pour assister le travail. » [1]
Design walkthrough	« Un « walkthrough » est un examen d'un produit par des pairs : les participants, qui sont à peu près au même niveau dans l'organisation, se réunissent pour examiner et discuter systématiquement une partie d'un logiciel. Ici, nous nous intéressons à la conception du logiciel du point de vue de l'utilisateur. » [1]
Utilisabilité	« L'utilisabilité (ou plus récemment--et facilement--usabilité[1]) est une notion proche de celle d'ergonomie. La norme ISO 9241 la définit comme « le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié ». » *Extrait de la définition sur Wikipedia{fr}*

Bibliographie

[1] MACKAY, Wendy E. (2006). *Cours de Conception et évaluation des Interfaces Hommes-Machine*.

MONJO, Frédéric (2007). Wikipedia{fr}. *Scrum*, [En ligne]. URL : http://fr.wikipedia.org/wiki/Scrum

CONVERSY, Stéphane (2006). *Cours de Conception participative*.

PATTON, Jeff (2006). *Interaction design meets Agility, Practicing Usage-Centered Design in an Agile software environment*.

MCINERNEY, Paul et MAURER, Frank (2005). Interactions magazine. *UCD in Agile projects: dream team or odd couple?*

AMBLER, Scott W. (2006). Agile Modeling. *Introduction to Agile Usability, User Experience activities on Agile Development Projects*.