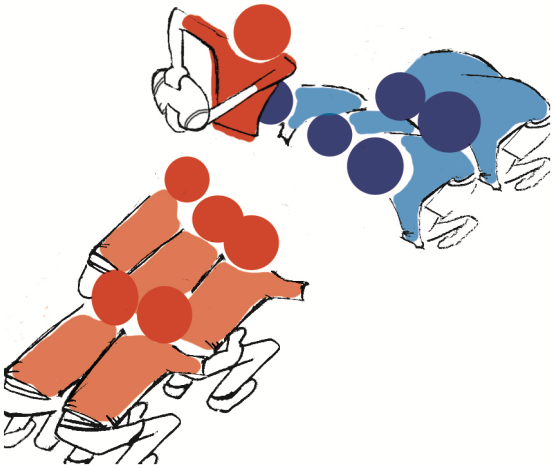
Thierry Cros

Spécifiez agile

Expression de besoins : la boite à outils du Product Owner



Être Agile

Spécifiez agile

Expression de besoins : la boite à outils du Product Owner

Thierry Cros

This book is for sale at
http://leanpub.com/agile-expression-de-besoins

This version was published on 2013-04-15

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book !

Please help Thierry Cros by spreading the word about this book on Twitter !

The suggested tweet for this book is :

Je viens d'acheter Spécifiez agile #specagile http ://tinyurl.com/SpecAgile

The suggested hashtag for this book is #specagile.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter :

https://twitter.com/search/#specagile

Table des matières

Préface

Spécifier et Agile sont des mots qu'on ne rencontre pas souvent ensemble.
Dans l'histoire du développement de logiciel, on a vu passer toutes sortes de spécification : spécification des besoins, spécification des exigences, spécification interne, externe, générale, détaillée… Dans les années 90, au plus fort de la vague de l'orienté objet, on a même vu apparaître une spécification "objet".
Mais de la spécification Agile, on n'en a pas entendu parler, à tel point qu'on se demande s'il ne s'agit pas d'un oxymore.
Une raison est probablement le BRUF (Big Requirements Up Front), autrement dit la grosse spécification détaillée élaborée au début, que tous les zélateurs de l'Agilité rejettent vigoureusement. A juste titre.
Cependant, et comme Thierry le démontre avec brio, ce n'est qu'un oxymore apparent : la petite spécification faite au bon moment, ça c'est typiquement Agile.
Dans "Spécifiez Agile", le travail de spécification dont il est question est relatif au "quoi", il a pour objectif de définir le comportement attendu du logiciel.
Ceux qui spécifient déjà, mais de manière traditionnelle (dans tradition, j'englobe largement : exigences, modèles, traçabilité, use-cases…) trouveront dans le livre de Thierry la preuve qu'on peut le faire différemment, avec les bénéfices de l'Agilité, et sans y perdre de rigueur.
Les équipes qui ont l'habitude de coder sans bien savoir quel est le besoin des utilisateurs y découvriront une approche légère,

avec les stories et leurs tests d'acceptation, qui les poussera à développer des produits plus alignés avec la "valeur métier".
Cet ouvrage est aussi dédié aux équipes déjà passées à l'Agilité (en particulier les Product Owners), auxquelles il propose de nouveaux outils pour aller de la vision au backlog.

Thierry vous invite à partager son enthousiasme pour "Spécifiez agile". Avec l'impératif du titre, c'est même plus qu'une invitation, c'est une injonction. Mais vous pouvez-la suivre en toute confiance.

Claude Aubry

Avant-propos

Voici l'histoire de "Spécifiez agile".

L'écriture émergente

Le Lean Startup est une proposition dont l'objectif est de :

> "Changer la façon de créer des entreprises, de lancer de nouveaux produits." Eric Ries

L'outil essentiel du Lean Startup, le "Minimum Viable Product" (MVP), fait écho aux premières versions d'un plan agile, basées sur la stratégie "Minimum Marketable Features" (MMF). Tout comme en agilité, il s'agit d'obtenir rapidement un feedback concret - d'une ou plusieurs versions minimalistes - afin de rectifier si nécessaire le produit. En pratique, ce feedback permet de *pivoter* ou *perséverer* :

- Perséverer dans la direction testée,
- "Pivoter", c'est-à-dire changer d'hypothèse, abandonner celle qui se révèle inefficace, qui ne rencontre pas ses Utilisateurs, pour repartir sur une nouvelle voie et donc un nouveau MVP afin de tester cette nouvelle hypothèse. C'est ce que permet le site Leanpub[1] dans le cas d'un *ebook* : publier un ouvrage électronique selon une voie lean startup.

[1] https://leanpub.com

Expression de besoins agile

Tout a commencé par l'idée d'un nouvel ouvrage consacré à l'expression de besoins agile.
Premier feedback : à l'issue de discussions avec des personnes reconnues dans la communauté agile, l'idée de l'ouvrage semble toujours d'actualité. L'approche agile est désormais perçue en tant qu'alternative viable. Plusieurs livres existent qui décrivent Scrum, Kanban et d'autres méthodes agiles. Cette richesse d'approches crée aujourd'hui une boite à outils de plusieurs centaines de pratiques. Il devient pertinent *a priori* de rédiger un ouvrage spécifique à un domaine, en l'occurence l'expression de besoins.
Deuxième étape : les Relecteurs sont très efficaces : ils n'hésitent pas à renvoyer tous les feedbacks qu'ils jugent utiles.
Tout cela pour aboutir à un véritable **premier MVP** (Minimum Viable Product) : la version initiale de l'ouvrage, au format électronique (pdf, epub, mobi). L'écriture continue alors que cette première version est publiée sur le site LeanPub. Les retours devraient permettre de consolider un contenu destiné à être édité au format "papier". Ce dernier est beaucoup moins malléable, comparé à un fichier "pdf" ou "epub".

Errare humanum est

J'étais très excité, à la fin des années 90, par deux nouveautés :

- Unified Modeling Language (UML),
- Extreme Programming (XP).

Plus le temps passait, plus XP, cette approche alternative dite "légère"[2] par opposition à la lourdeur des méthodes telles que le processus unifié ou le cycle en V, m'attirait. Je me lançais (en 2003) dans l'écriture d'un livre sur l'Extreme Programming. Le "gang des quatre" XPistes - Jean-Louis Bénard, Laurent Bossavit, Régis Médina, Dominic Williams - avait publié avec succès en 2002, chez Eyrolles, le livre de référence sur XP[3].
Le mien - *Maîtrisez les projets avec l'Extreme Programming* - fut un fiasco.

Pourquoi raconter cette histoire

Ce premier livre publié en 2004 était le résultat d'une obstination. Je n'avais pas sollicité de feedback concret, rapide. Je n'avais pas collaboré. Je n'en avais pas eu le courage.

> Errare humanum est, perseverare diabolicum[4]

Si l'agilité donne droit à l'erreur, nous avons le devoir - en contrepartie - de nous améliorer. L'histoire de ces deux livres (sans préjuger du destin de "Spécifiez agile") pourrait être une métaphore : agiliser vos développements de produits complexes grâce à plus de :

- communication,

[2] Les méthodes alternatives, empiriques, des années 90 étaient qualifiées de *légères* avant de devenir *agiles* après la publication du "Manifeste agile" en 2001.

[3] Ce livre est disponible (format électronique) gratuitement ici : Gestion de projet eXtreme Programming au format PDF sans DRM.

[4] « Se tromper est humain ; persévérer est diabolique ».

- feedback concret et rapide,
- simplicité,
- courage.

Historique

15 avril 2013
Quelques modifications complémentaires, en particulier dans les chapitres “Intelligence collective” et “Valeur métier”.

10 avril 2013
Refonte du plan (voir ci-dessous : “conception de l’ouvrage”), prise en compte de feedbacks de lecteurs (pratiquement dans tous les chapitres), ajout du chapitre “intelligence collective”.

9 janvier 2013
Prise en compte de feedbacks de lecteurs sur l’ensemble des chapitres,
consolidation des études de cas pour insister sur le côté *pratique* et la démarche ascendante,
modification du chapitre “harmonie” qui devient “démarche” et précise l’intérêt d’une démarche ascendante de spécification
couverture : changement de police.

18 décembre 2012
Ajout du glossaire et de la bibliographie,
Quelques typos corrigées,
Compléments mineurs dans plusieurs chapitres.

17 décembre 2012
Première version mise en vente.

Objectifs de “Spécifiez agile”

L’objectif de cet ouvrage est de fournir un guide des pratiques d’expression de besoins agile. Il est le compagnon du Product Owner, de toute personne amenée à spécifier un produit complexe destiné à être développé dans un cadre agile.
Les ScrumMasters et Coaches agiles y trouveront une synthèse de pratiques connues et quelques nouveautés.

Conception de l’ouvrage

Les premiers feedbacks de Lecteurs m’ont conduit à modifier sensiblement l’ordre des chapitres.

Un autre monde

L’agile est si différent ! Avant de parcourir les différents concepts et pratiques, autant savoir “où on met les pieds”. Et quoi de mieux que de raconter une histoire de spécification agile ? C’est le but du chapitre “Le Club des Chercheurs de Trésors”. Lisez absolument ce chapitre si

- vous ne connaissez pas les concepts d’expression de besoins agile (stories…)
- vous connaissez ces concepts mais vous ne voyez pas très bien ce que peut bien vouloir dire “émergence des spécifications” (qui est un principe agile).

Concepts et pratiques

Le cadre agile est posé. Voyons maintenant en détail les concepts et pratiques utiles à un Spécifieur.

Être agile

Les concepts et pratiques sont spécifiques à l'agilité, la *démarche* est radicalement différente. Plutôt qu'une approche strictement descendante (du général vers le particulier), nous faisons *émerger* les spécifications, du particulier au général. Cette partie propose également une "discipline" d'intelligence collective, basée sur la *gestion par consentement* de la sociocratie. En effet, une équipe auto-organisée offre de meilleurs spécifications (c'est un principe agile) pourvu qu'elle s'arme de pratiques pertinentes.

Modéliser

Les concepts présentés jusqu'à présent ne sont pas nécessairement adaptés à toutes les personnes ni à tous les produits. C'est le cas de systèmes peu ou pas interactifs. Rien n'empêche - en agilité - de modéliser. Cette partie de l'ouvrage est consacrée à ce type d'approche et propose une étude de cas.

Conclusion

Quelques mots de conclusion - à vous de jouer ! - une bibliographie (courte !) et un glossaire pour terminer l'ouvrage.

Remerciements

Remercier les Relecteurs de “Spécifiez agile” est un vrai bonheur. J’ai été très touché par la qualité de leur feedback, de leur engagement.

Merci à Caroline Damour qui est Product Owner et responsable de port-folio ; ton point de vue “opérationnel” et les suggestions que tu me proposais sous forme de questions m’ont aidé à rester “centré Utilisateur”.
Émilie Franchomme, m’a offert de nombreux commentaires, aussi bien sur le fond que sur la forme de “Spécifiez agile”. Merci Émilie, j’ai vraiment apprécié tes suggestions, tes reformulations, bien plus pertinentes que ce que je proposais.
Alfred Almendra est une source inépuisable de ressources agiles. Ses propositions d’amélioration furent très appropriées tout au long de l’écriture, y compris la conclusion de l’ouvrage. Merci Alfred, je suis vraiment heureux que tu aies participé à cette aventure.
Claude Aubry, je me souviendrai de nos discussions autour d’un café, elles m’auront permis de mieux exprimer la rupture agile : besoin et solution sont intimement liés. Merci pour ces belles discussions, tes retours d’une grande pertinence sur le fond de ce que je souhaitais exprimer (et ausi quelques typos). Tes feedbacks ont été copieux. Merci également pour la préface, c’est un exercice qui n’est pas si facile.

Merci à Alexandre Boutin et Jean-François Jagodzinski dont les feedbacks m’ont secoué. Que ce soit sur la question de l’évolution du système d’information ou plus généralement sur le ton de l’ouvrage, votre apport a été salutaire.

Je remercie Oana Juncu, David Brocard, Fabrice Aimetti, Romain Couturier et Laurent Carbonnaux. Je vous ai demandé, tardivement, de rejoindre cette aventure et vous avez été bienveillants, le feedback que vous m'avez renvoyé a contribué à l'ouvrage.

Merci à Laurent Morisseau, également membre fondateur de la fédération agile.

Merci à Sébastien Cros. Il est l'auteur de l'illustration de couverture et des portraits des *personas* du "Club des Chercheurs de Trésors". Sébastien a su traduire visuellement ce que je souhaitais, ce qui était loin d'être évident.

Enfin, je souhaite remercier Christine Julio, Brigitte Bourbée et Bruno Bourbonnais de la Maison de l'Initiative[5], la "société coopérative et participative" (SCOP) dans laquelle j'ai le statut d'entrepreneur-salarié. Leur soutien simple, professionnel, humaniste me fait chaud au coeur. Je suis fier de faire partie de cette équipe.
La Maison de l'Initiative est une CAE - Coopérative d'Activité et d'Emploi - qui réunit plusieurs dizaines de personnes désireuses d'être "entrepreneur" tout en faisant partie d'une organisation de l'Économie Sociale et Solidaire.

Castans,
le 18 décembre 2012.

[5] http://www.maison-initiative.org/?-L-equipe-d-appui-

Un autre monde

1 Introduction

Traditionnellement, le développement d'un produit distingue deux mondes :

- l'espace du besoin
- et celui de la solution.

Des documents écrits - par le Maître d'Ouvrage (MOA) - expriment le besoin : Cahier des charges fonctionnel[1] et Spécification technique de besoin[2]. Leur contenu est normalisé (AFNOR...).
Les dossiers de spécification - spécifications fonctionnelles, spécifications techniques, spécifications d'architecture - sont la traduction des besoins : ils "répondent" aux Cahier de Charges.
Les spécifications sont alors transformées en produit par le Maître d'Oeuvre (MOE) : la solution technique est ainsi décrite dans un "dossier de conception". Ces différents documents sont parfois illustrés grâce à une modélisation UML (Unified Modeling Language).

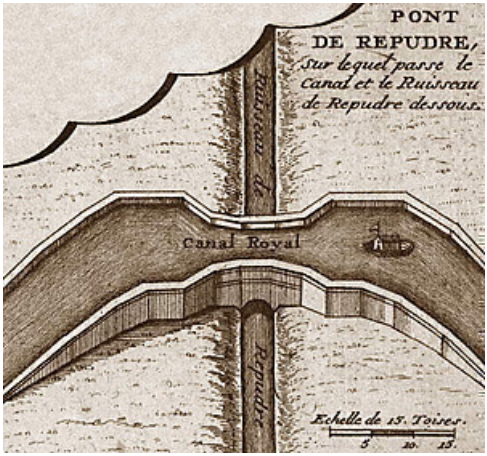
Ces rôles (MOA, MOE) et pratiques (élaboration de gros documents) sont vieux de plusieurs siècles. Leur origine remonte au Moyen-Age. Les ponts[3] devaient faire face à de lourdes... charges[4].

[1] Cahier des charges fonctionnel

[2] Spécification technique de besoin.

[3] Image par Peter Gugerell. "Pont de Repudre, sur lequel passe le Canal et le Ruisseau de Repudre dessous" Canal du Midi.

[4] Voir cette histoire des ponts.



Le pont-canal de Répudre (Canal du Midi)

Le Manifeste agile, en 2001, est une évolution radicale : les spécifications et conceptions *émergent*. Elles s'élaborent en équipe et prennent en compte un feedback (retour d'information) concret et rapide : celui des mises en exploitation fréquentes du produit.

1.1 Terminologie

Dans ce livre, **expression de besoins** représente *tout* ce que le Client ou son représentant *extériorise* - document, discussion - afin que l'équipe de réalisation fabrique le produit le plus adapté à sa demande.

La **spécification** est l'activité qui consiste à exprimer le besoin.

1.2 Product Owner et Spécifieur

La spécification - le fait d'exprimer les besoins - est le rôle du **Spécifieur**. C'est l'une des activités essentielles du *Product Owner*[5] (PO), rôle emblématique de la méthode Scrum[6], devenu standard de fait agile.
Spécifieur correspond à ce rôle de Scrum.

1.2.1 Spécifieur

Spécifieur représente aussi

- le rôle de *Product Manager* d'autres méthodes agiles (Extreme Programming, Lean Software Development[7])
- plus généralement, toute personne participant à la *spécification* d'un produit :
  - Analyste fonctionnel
  - Expert métier
  - Marketeur
  - Consultant Assistance Maîtrise d'Ouvrage
  - Testeurs fonctionnels
  - Chef de Projet Utilisateur
  - Chef de Projet fonctionnel.

[5]La planification et la validation sont les autres activités importantes du Product Owner.

[6]Pour découvrir Scrum : Guide Scrum. Voir aussi le livre de référence de Claude Aubry (cf bibliographie en fin d'ouvrage).

[7]Pour des informations sur ces méthodes agiles, voir les articles de mon site http ://thierrycros.net. Voir aussi Lean Software Development.

1.3 Exigences ?

Le terme “exigence” désigne un pré-requis à la conception et la réalisation d’un produit. Selon Wikipedia :

> En ingénierie, et plus particulièrement dans les procédures d’appel d’offres publiques et privées, les exigences sont l’expression d’un besoin documenté sur ce qu’un produit ou un service particuliers devraient être ou faire.

La planification agile repose sur la capacité à prioriser (ou prioritiser) les besoins - essentiellement en fonction de leur valeur métier[8] - et développer uniquement ce qui apporte effectivement une valeur aux Utilisateurs[9]. Cela signifie que certains besoins ne seront pas - au final - implémentés. Le *terme* d’exigence, qui sous-entend une obligation, est par conséquent inadapté à l’expression de besoins agile.

1.4 Sur le terrain

La métaphore “rugby” est naturelle[10] en agilité (*Scrum* signifie “mêlée”). Mais… Quelle analogie entre rôle Scrum et numéro de

[8] La priorité d’un besoin est fonction de la valeur métier et d’autres critères, par exemple la diminution de risque, la complexité, la pertinence. Les principes agiles insistent sur la valeur métier : Manifeste agile.

[9] La Valeur d’un besoin dépend de ce qu’il apporte à l’Utilisateur. Ce terme représente toute personne qui utilise le produit logiciel, quel que soit son rôle.

[10] L’illustration du livre de référence, de Claude Aubry, est une photo de rugby.

maillot ?
Voici un exemple de correspondance.

Rôle Scrum	Rugby
Product Owner	Demi de mélée (numéro 9)
ScrumMaster	Capitaine, Arrière…
Développeur	Pack (mélée)

Le Product Owner - demi de mêlée - est un joueur sur le terrain. C'est la pratique "Client sur Site" de l'Extreme Programming. Autrement dit, il "mouille le maillot" tout comme les autres joueurs. C'est le sens de l'illustration de couverture.
Le demi de mêlée est un "rouge" ; les "bleus" - adversaires - représentent tous les obstacles, les *impediments* qui viennent contrarier l'avancement de l'équipe complète.

Jean-Marc Doussain : behind the scrum

Cette photo[11] de Pierre Selim[12] est complémentaire de l'illustration de couverture. Ici, le demi de mêlée (le 9) est derrière le pack pour récupérer le ballon qu'il avait au préalable envoyé au milieu dans la mêlée.

Sur le terrain, chaque joueur joue un rôle précis : demi de mêlée,

[11] photo sous licence Creative Commons Paternité 3.0.
[12] https://fr.wikipedia.org/wiki/Fichier:Jean-Marc_Doussain_behind_the_scrum.jpg

pilier… Il est aussi “multi-disciplinaire” : en phase défensive, chaque joueur “redescend” pour participer à la défense, même si ce n’est pas sa fonction habituelle. Ce qui tord le cou à une croyance très répandue qui consiste à confondre efficacité d’une équipe et efficacité des équipiers.

1.4.1 Product Owner, qui es-tu ?

Le Product Owner est le *leader* de l’équipe.
En tant qu’Agiliste :

- il aime communiquer, il sait lire, écrire, écouter, parler ;
- il recherche et sollicite du feedback, il sait recevoir ce feedback ;
- il choisit la solution la plus simple ;
- il s’améliore ;
- il est présent et engagé.

Le Product Owner

- connaît le processus métier dans lequel s’inscrit le produit dont il est responsable ;
- sait dans quel contexte s’inscrit le développement de ce produit, il aura à déterminer sa valeur métier, elle-même fonction de la valeur stratégique ;
- traduit les objectifs métier en caractéristiques du produit ;
- fait des choix de priorité en fonction de la valeur métier ;
- sait qu’il vaut mieux mettre en exploitation un petit incrément du produit.

Lorsque le Product Owner est issu du domaine technique du produit, il a tout intérêt à se faire aider de personnes qui connaissent le métier : Utilisateur expert par exemple. À l'inverse, un Product Owner qui ne connait pas la technique peut être épaulé par un Analyste fonctionnel.

1.5 Communauté agile

Que ce soit la question de l'expression de besoins ou tout autre sujet en relation à l'agilité, la communauté agile *locale* est une excellente source d'inspiration. Des rencontres entre praticiens - Spécifieurs, Développeurs, Coaches… - sont organisées, de plus en plus au format "open" (forum ouvert), c'est-à-dire qu'il n'y a pas de sujet pré-établi. Une bonne opportunité pour en proposer.

2 Les niveaux d'expression de besoins agile

L'expression de besoins agile se situe à plusieurs niveaux, plus ou moins détaillés. À chaque niveau correspond une planification qui est une *projection dans le temps* des besoins exprimés.

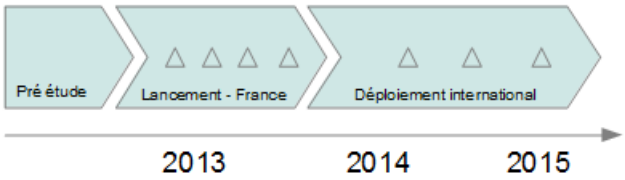
2.1 La vie du produit

Si la durée de vie d'un logiciel est extrêmement variable, de quelques minutes à plusieurs décennies, elle s'établit généralement à plusieurs années. L'expression de besoins agile s'inscrit dans cette *chronologie*, constituée de différentes phases. Les phases du produit constituent sa **feuille de route**, pluri-annuelle. Une phase est planifiée en **versions**, elles-mêmes découpées en **itérations**.

Français	Anglais
Feuille de route	Roadmap
Version	Release

2.1.1 Durée de vie : plusieurs années

Voici un exemple de feuille de route de produit. Chaque phase (de produit) contient plusieurs versions.



Phases et versions

Ici, la phase de lancement peut être vue comme une période dans laquelle les sites concernés sont "pilotes". Cette phase est constituée de quatre versions dans ce schéma. La planification des phases dépend de nombreux facteurs : sites et personnes concernés, fonctionnalités, contraintes réglementaires… Ce sont les *objectifs* du produit[1] qui guident la mise au point de ses différentes phases. Cette projection des objectifs dans le temps dépend entr'autres du feedback concret obtenu grâce à l'utilisation du logiciel.

2.1.2 Projet et maintenance

Une phase de produit se concrétise généralement en périodes alternées de "projet" et "maintenance". Ce découpage est fonction de l'organisation : budget seuil, équipes… Du point de vue de l'Utilisateur, une nouvelle mise en exploitation (nouvelle version ou correctif) constitue un changement du produit, que ce changement soit obtenu par "projet" ou pas.
Cette dichotomie - solidement ancrée dans les organisations - est

[1]Le terme "objectif" correspond à l'un des concepts présentés ici : évolution, objectif, feature, user story.

due à la primauté de règles comptables[2], artificielles en regard de la vie du produit et de son usage.

2.2 Plusieurs niveaux de plans

Voyons dans un premier temps les différents niveaux de plans. Nous en déduirons alors les niveaux d'expression de besoins.

2.2.1 Schéma directeur

Ces différents niveaux (feuille de route, plan de versions, plan d'itérations) s'inscrivent plus généralement dans le *plan directeur* du système d'information[3]. Et à ce niveau, ce sont des **évolutions du Système d'Information** (SI) qui constituent les éléments d'expression de besoins.

L'équivalent d'un schéma directeur chez un éditeur de logiciels est le plan pluri-annuel de la gamme de produits.

2.2.2 Phase Produit

Chaque phase (le "phasing" du produit) s'inscrit dans une étape du cycle de vie du produit[4]. Plus précisément, une phase est liée à des *objectifs* assignés au logiciel, en conformité avec les enjeux de l'organisation.
Ici, il s'agit de phase du produit, pas de phase d'un cycle projet. Le terme "phasing" représente une phase produit dans la suite du

[2]Voir par exemple Immobilisation et amortissement
[3]Schéma directeur informatique, journal du net)
[4]Voir par exemple Le cycle de vie d'un produit.

chapitre.
Un phasing peut être caractérisé également par un déploiement spécifique. Par exemple un produit déployé pour une population donnée puis plus tard adapté à d'autres types d'Utilisateurs. Un phasing est constitué d'une ou plusieurs versions et répond à des objectifs spécifiques.

2.2.3 Version

Une version est constituée de caractéristiques (les *features*) fonctionnelles et non fonctionnelles. D'une version à l'autre, il y a ajout de fonctions (aspect incrémental du produit) ou bien raffinement d'une fonction existante (aspect itératif du développement de la fonction). Une version est constituée d'une ou plusieurs itérations ou cycles de développement.

Le terme "version" recouvre donc deux définitions :

- Version au sens "produit mis en exploitation", la version caractérise le contenu du produit
- Version au sens "boite de temps" qui permet de fabriquer le contenu ci-dessus.

2.2.4 Itération

Cette petite "boite de temps" permet un feedback plus concret et rapide du développement. Elle a pour objectif la finalisation de petites interactions (les stories), selon la *définition de fini* (l'ensemble des critères qui permettent de déclarer "terminé" un développement). Autrement dit, une itération produit un

incrément du produit.
Notons tout de suite qu'une itération offre plus qu'un nouvelle version du produit : également une nouvelle version des différents *artefacts* de pilotage (backlog du produit…).

Basée sur une pratique d'ingénierie capitale : l'intégration continue, l'itération permet de détecter rapidement d'éventuels défauts et donc de les rectifier "à temps". L'intégration continue est essentielle car elle autorise le passage systématique de tests d'acceptation, qu'ils soient automatisés ou non. Autrement dit, l'objectif est d'éviter un stock trop important de logiciel non intégré, non validé donc potentiellement entâché de défauts.

2.3 Proposition de niveaux d'expression de besoins

Chaque temporalité du produit logiciel se caractérise par un niveau d'expression de besoins plus ou moins détaillée.

Un élément	**est projeté en**	**dans**	**Rythme**
Évolution SI	Projet	Schéma directeur	Annuel/Semestriel
Objectif (vision)	Phasing	Feuille de route	Semestriel/Trimestriel
Feature	Version	Plan de versions	Trimestriel/Mensuel
Story	Itération	Plan d'itérations	Hebdomadaire/Journalier

Le rythme de chaque élément est adapté en fonction du contexte de l'organisation.
Il existe un cinquième niveau de planification, à l'intérieur de l'itération, qui est celui des tâches de réalisation. Ce niveau est planifié en termes d'actions de réalisation telles que conception,

programmation, tests et non pas en terme d'expression de besoins[5]. L'objectif principal de la réunion quotidienne - Stand Up Meeting ou Daily Scrum Meeting - est l'auto-organisation de l'équipe dans la réalisation de ces tâches.

2.3.1 Évolution du Système d'Information

Une évolution du Système d'Information (S.I.) peut être

- Le déploiement d'une nouvelle solution,
- L'amélioration ou adaptation d'une solution existante,
- Le retrait d'une solution (qui peut donner lieu à archivage des données par exemple).

Ces évolutions se concrétisent en *programmes* (ensemble de projets liés), *projets* ou *activités de maintenance* selon leur importance.

2.3.2 Les objectifs assignés au produit

La vision d'un produit (attention à ne pas confondre avec une charte projet[6]) permet de situer ce produit dans l'organisation et surtout lui donne un sens. Elle devrait fournir le contexte, les **objectifs**, les principales caractéristiques du produit logiciel, de façon concise. Autrement dit, la vision exprime **la raison d'être** du produit. La charte projet est spécifique du projet, pas

[5] Ne faisant pas partie de l'expression de besoins, les tâches ne sont pas étudiées dans ce livre.

[6] Voir par exemple Charte de projet.

du produit.
Les objectifs sont qualitatifs et/ou quantitatifs. Dans tous les cas, ils devraient être munis d'indicateurs permettant de vérifier qu'ils ont été - ou non - atteints.

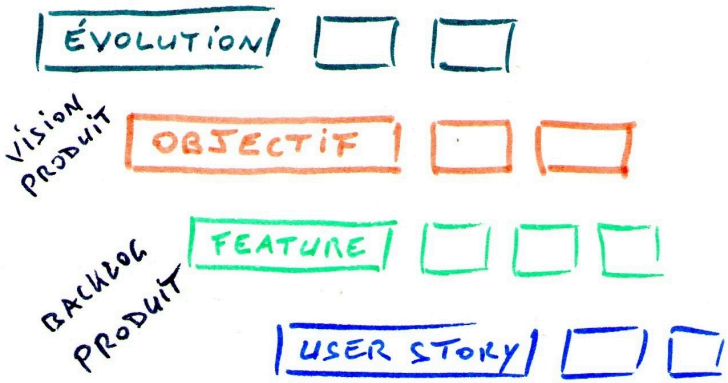
2.3.3 Feature

Une *feature* ou caractéristique du produit a du sens pour un Utilisateur et lui apporte plus ou moins de valeur. Concrètement, une feature est une fonctionnalité ou bien un besoin non fonctionnel typique (par exemple : être disponible sur SmartPhone ou bien précision d'un calcul).
Une feature est liée à la version dans laquelle elle est implémentée et mise en exploitation. À ce titre, est est une nouveauté de la version (produit incrémental) ou bien un raffinage d'une fonctionnalité déjà présente dans le produit (itératif sur cette feature).

2.3.4 User Story

La user story est probablement l'élément d'expression de besoins agile le plus connu. Elle constitue - munie de ses tests d'acceptation - le niveau le plus précis. Un format classique est

```
En tant que [rôle] je peux [interaction] afin de
[bénéfice attendu].
```



Niveaux d'expression de besoins

Un **backlog** est une liste (généralement ordonnée selon un certain critère) de choses à faire (qui ont du sens pour l'Utilisateur). Dans ce schéma, une *évolution* (dans le backlog d'évolutions) est raffinée en *objectifs* dans la "vision" du produit correspondant. Ces objectifs sont implémentés en caractéristiques ou *features*. Une feature est détaillée en *stories*. Features et stories constituent le backlog du produit.
Un backlog est constitué de quelques dizaines d'éléments maximum afin de respecter le principe agile "peu de stock".

2.4 Similarités entre niveaux

2.4.1 Backlog

Chaque élément fait partie d'une "liste des choses à faire".

Nous devrions donc obtenir :

- Le backlog d'évolutions, au niveau du portefeuille du service ou de la DSI,
- Le backlog d'objectifs associé à la vision du produit qui implémente l'évolution,
- Le backlog de caractéristiques - features - qui raffinent la vision,
- Le backlog de user stories qui raffinent une feature (features et stories pouvant être consignées dans le même backlog).

Si un élément de niveau plus détaillé "raffine" un niveau plus grossier, la démarche suivie peut être tout autant descendante (du général au particulier) qu'ascendante.

2.4.2 Attributs de planification

Quel que soit son niveau, un besoin est qualifié par des attributs qui permettent de le planifier. En particulier :

Attribut "Valeur Métier"
La valeur métier est l'importance, l'intérêt qu'a ce besoin, du point de vue de l'Utilisateur. De même que le coût, cette valeur peut être exprimée en plusieurs unités : en euros ou bien en points de valeur relatifs. Voir le chapitre Valeur métier.

Attribut "Coût"
Qu'il soit exprimé en euros, en jours, semaines ou mois ou bien encore en points d'effort ou de difficulté relatifs, le coût est

un élément nécessaire à la planification[7]. L'effort ne représente pas uniquement la *complexité*. La "volumétrie" du besoin est également prise en compte : une information comportant vingt données simples demande plus d'effort qu'une information ne comportant que cinq données.

Voir le paragraphe "Planification agile" pour plus d'informations.

2.4.3 Juste à temps

Chaque élément d'expression de besoins suit le principe : **précis à court terme**, **grossier à long terme**. Ce qui correspond au principe Lean de "Juste à temps"[8]. Attention : "imprécis à long terme" ne signifie pas "inutile". Il peut être utile de définir des éléments grossiers à plus long terme (par exemple quelques grosses stories pour la fin d'une release), cela donne "le cap" de la release. Ainsi, des informations en relation avec ce cap seront retenues. Sans cela, elles passeraient inaperçues car elles ne seraient pas jugées pertinentes. De plus, ces éléments grossiers permettent d'estimer le "reste à faire".

2.4.4 Niveaux intermédiaires

Selon sa maturité, un élément d'expression de besoins est non seulement grossier, il est aussi trop "lourd" en terme de coût pour être raisonnablement planifié. Le besoin étant exprimé "juste à

[7] Toutefois, certaines équipes assignent simplement un coût de 1 à chaque story et la vélocité est alors le nombre de stories qu'il est possible de terminer dans un cycle ou itération. Ce sont des stories équivalentes en termes d'effort. Plus précisément, les différences d'effort sont "absorbées" par la quantité d'éléments.

[8] Lean.

temps", il peut être légitime de conserver des éléments sans les préciser davantage.

Les termes suivants sont utilisés dans ce livre pour décrire ces éléments qui nécessiteront un découpage plus fin.

- Thème : c'est une feature plus lourde que les autres, voire un ensemble de features,
- Epic : idem au niveau d'une story.

Ces termes ne sont pas (à ce jour : fin 2012) standardisés. D'autres définitions peuvent donc exister, par exemple "epic" pour une grosse feature. Toutefois, ce terme d'epic existe pour désigner une grosse story depuis plusieurs années.

3 Valeur métier

La valeur métier est une question fondamentale, en particulier en agilité : en quoi les produits gérés par la DSI contribuent-ils à l'organisation ? Comment la soutiennent-ils ? William Deming insiste sur la valeur métier en relation à la diminution des coûts.

> Dans les années 1970, la pensée de Deming a pu être résumée comme une alternative « a sinon b » :
>
> (a) Quand les gens et les organisations se concentrent sur la qualité, définie comme la satisfaction des besoins et des désirs des utilisateurs, la qualité augmente et les coûts chutent.
>
> (b) Sinon, quand les organisations se focalisent sur les coûts, la qualité tend à diminuer au cours du temps. Source Wikipedia[1]

L'informatique est désormais clairement mise à contribution : elle doit participer activement à l'atteinte des objectifs de l'organisation, elle doit être *alignée*.

> Les projets IT ne doivent plus seulement supporter l'activité, mais aussi l'optimiser et la dynamiser.

[1] Voir la page Wikipedia consacrée à Deming

> C'est désormais la "business value" qui prime dans les grandes orientations et choix du portefeuille d'applications. Journal du Net 30 Juin 2008[2]

Une user story consolide la valeur métier d'une feature, qui à son tour contribue à la valeur du produit. Et ce dernier participe à la création de valeur de l'organisation.

3.1 Qu'est-ce que la valeur métier ?

Un développement de produit n'est jamais "gratuit". Que ce soit en euros et/ou en temps, il a un coût. La valeur métier est ce qui donne du sens à ce coût. Il est alors possible d'établir le ratio valeur/coût, le retour sur investissement.
Simplement énoncée, la valeur métier est l'intérêt qu'a l'Utilisateur à disposer de ce nouveau produit.

3.1.1 Valeur métier : les facteurs de valeur

Voici une représentation de facteurs de valeur, de caractéristiques qui contribuent à la valeur : le nid d'abeille de Peter Morville[3].

[2] Voir l'article Définir et évaluer l'apport métier des applications
[3] Schéma reproduit avec son autorisation.

Utilisabilité : nid d'abeille de Peter Morville

Dans *User Experience Design*[4], Peter Morville explique les différents facteurs de la valeur, pondérés en fonction du contexte du produit.

- Utile : au-delà de ce qui est attendu du management, le produit est-il réellement utile ?
- Désirable : quel est l'attirance de l'application ? Son image ?
- Accessible : quelle est l'accessibilité du site, du logiciel, en particulier pour les déficients visuels ?
- Crédible : en quoi la conception du produit contribue-t-elle à la crédibilité du logiciel[5] ?

[4]Voir User Experience Design
[5]Voir Web Credibility Project

- Atteignable : le produit est-il facilement trouvé dans le système d'information, sur Internet ?
- Utilisable : le produit est-il facile à utiliser ?

Cette modélisation recentre la valeur : ce sont les **Utilisateurs** qui la définissent[6].

Au-delà du produit lui-même - utilité, accessibilité… - d'autres critères complètent l'expérience Utilisateur.

- Performance : quid des temps de réponse ? De la précision des résultats ?
- Niveau de service : le "Help Desk" est-il performant ? La reprise en cas d'incident est-elle suffisamment rapide et fiable ?

3.1.2 Impliquer l'Utilisateur, le vrai

La valeur métier ne se décrète pas par procuration : tel Consultant ou tel Manager est probablement pertinent dans ses propositions. Il reste que c'est l'Utilisateur, le vrai, qui détient les clés de la valeur effective. L'utilité dépend de la place réelle du produit dans son processus métier.
En quoi ce produit facilite le processus, en quoi il permet ce qui était impossible auparavant… Autant de questions qui aident à "cerner" la valeur métier.
D'où l'importance, dans une approche agile, de travailler avec de véritables (futurs) Utilisateurs du produit. D'autant plus que cette implication au plus tôt facilite la conduite du changement

[6] Le site usability.gov propose de nombreuses pratiques en relation à la valeur métier.

induite par le nouvel outil informatique.
Plutôt que de rechercher des formules compliquées, travailler directement avec l'Utilisateur est un moyen *simple* de valoriser le logiciel. Valeur métier : faites simple. Adoptez le slogan :
KISS[7] **your Business Value**.

Jouer pour apprendre la valeur métier
Le jeu "Business Value Modeling" du site The Agile Coach Tookit[8] est une bonne introduction à la valeur métier en agilité et peut être joué avec les Spécifieurs qui auront à estimer les valeurs des features.

3.1.3 Intérêt des Utilisateurs et valeur stratégique

De façon générale, la valeur métier d'un produit est fonction

- d'une part de l'intérêt de ce produit pouir ses Utilisateurs
- de la valeur stratégique du produit pour les Managers.

Le Product Owner, garant de la valeur métier, est parfois pris entre deux feus, lorsque les intérêts des Utilisateurs et du Management divergent.

3.1.4 Temps et valeur

Le temps est une composante de la valeur. Généralement, l'estimation - même relative - de la valeur suppose qu'elle est fournie

[7] KISS pour Keep It Simple, Stupid ! principe KISS.
[8] http://www.agilecoach.net/coach-tools/business-value-modeling/

au bon moment, “juste à temps”.
Une livraison trop tardive diminue, voire supprime, la valeur escomptée au départ[9].

3.2 Trois pratiques simples

3.2.1 Déterminer les valeurs métier relatives

La valeur est particulièrement utile au niveau du backlog d’évolutions afin d’établir la liste des projets effectivement lancés. Elle a son importance, dans un projet, au niveau des features qui sont projetées dans le temps sous forme de versions successives du produit.

Voici une pratique simple, adaptée du Planning Poker[10] : **Business Value Poker**[11].

1. Obtenir la liste des éléments à pondérer
2. Déterminer l’élément le plus important en terme de valeur métier (la référence)
3. Lui attribuer arbitrairement une valeur métier de 1000
4. Pondérer ensuite chaque élément restant en fonction de sa valeur, relativement à la référence
5. Appliquer ensuite la règle : pas 2 fois la même valeur métier.

[9] Incorporating cost of delay in feature prioritization.
[10] Voir le référentiel des pratiques agiles : Planning Poker.
[11] Ce jeu est aussi connu sous le nom de Business Value Game.

Cette pratique se joue dans le cadre d'ateliers dont l'objectif est de déterminer collectivement les valeurs métier des éléments d'expression de besoins.

Le chapitre Features propose d'autres pratiques, en particulier des jeux innovants.

3.2.2 Investigation

Quelle que soit la méthode utilisée, la valeur métier supposée avant une mise en exploitation doit être confrontée à la réalité de l'utilisation.

Enquêtes de satisfaction
Une enquête de satisfaction reste un moyen utile de vérifier ces estimations de valeurs métier initiales. C'est une pratique simple qui permet de prendre conscience de la réalité du terrain et donc de rectifier si nécessaire les solutions mises à disposition des Utilisateurs. La *tendance* dans le temps permet de vérifier que les actions entreprises pour améliorer la valeur métier apportent effectivement un résultat, du point de vue de l'Utilisateur.

Cette enquête pourrait porter sur l'importance relative des features mises en production.

> Selon vous, quelles sont les fonctions du logiciel les plus importantes ? Notez les de 0 à 5 (5 : la plus importante).

Supposons une version mise en exploitation, qui contient les features F1 F2 F3 F4. Les estimations de valeurs relatives (lors de la planification de la version) ont établi un classement. L'enquête

permet alors de vérifier la perception des Utilisateurs.
L'estimation porte sur une échelle de 0 à 1000. L'enquête sur une échelle de 0 à 5.

Feature	Estimation	Enquête
F1	1000	5
F2	900	4
F3	800	2
F4	400	4

Ce tableau montre que le Product Owner avait vu juste à propos des deux premières features. Par contre, les Utilisateurs ont plus valorisé F4 que F3. Cela devrait se traduire par un changement de priorité dans les features à venir en relation à F3 et F4.

Les critères décrits plus haut - utilité, désirabilité, disponibilité… - peuvent être utilisés dans l'enquête afin de mieux comprendre les résultats. Par ailleurs, ces résultats d'enquête peuvent être confrontés à des statistiques d'usage.

La question de la valeur effective - en exploitation - se pose également au niveau plus global du produit dans l'organisation. La comparaison entre features d'un produit devient alors une analyse comparative de la valeur des produits d'un portfolio (portefeuille).

Une approche **empirique**, basée sur une expérimentation concrète du produit, permet - petit à petit - de mieux comprendre ce qu'est la valeur métier, dans un contexte donné, unique.

Le *Help Desk* est une excellente source d'informations concrètes[12]. Il "remonte" des données statistiques sur les défauts, les questions

[12]Une planification agile est basée sur des mises en exploitation fréquentes : le produit est exploité alors qu'il est encore en développement.

posées par rapport aux features du produit. Ces données sont alors précieuses lors de la planification des versions suivantes du produit.

Que demande l'infirmière ?
Contrairement à une idée reçue, une mesure n'a pas besoin d'être objective pour être pertinente. C'est le cas d'une mesure de la douleur : l'infirmière demande de noter la douleur sur une échelle de 0 à 10. Cette note est très subjective, néammoins elle a du sens : après administration d'une drogue adaptée, la douleur est sensée diminuer. C'est le *delta* entre les deux notes qui importe.
Ces notations subjectives de la valeur sont complémentaires. Elles consolident les mesures basées sur des informations plus objectives, par exemple "nombre de Visiteurs du site".

Go & See
Cette pratique issue du "Toyota Production System" - genba attitude[13] - consiste à être "au bon endroit" : chez l'Utilisateur. Être à l'endroit où le logiciel est effectivement utilisé aide à comprendre l'expérience Utilisateur et par conséquent ce qui lui apporte de la valeur. Il s'agit d'être **témoin vigilant** dans l'intention de détecter ce qui a véritablement un intérêt.

Lean Startup
Une démarche Lean Startup[14] basée sur des "Minimum Viable Products" (MVP) permet de cerner petit-à-petit la valeur métier. Il existe finalement peu de différence entre un plan agile, basé sur des versions "Minimum Marketable Features" (MMF) et le Lean Startup. Ce dernier insiste sur la capacité à *pivoter*, c'est-à-

[13]Voir Go & See et San gen shugi.

[14]Voir cette présentation slideshare d'Eric Ries : Building a Virtual World : Lean Startup Style

dire tester une nouvelle hyothèse pour améliorer la valeur métier. C'est exactement le rôle d'un du Product Owner agile.
Dans cet ordre d'idée, une enquête de satisfaction portant sur l'existant permet d'orienter le développement du nouveau produit.

3.2.3 Le jeu de la perfection

Une autre forme de questionnaire consiste à jouer le "jeu de la perfection" appliqué au logiciel[15].
La première étape consiste à déterminer la portée du questionnaire :

- Le logiciel (exclusivement)
- Le système (matériel et logiciel)
- Le service rendu par "l'informatique" (y compris Help Desk…)
- La solution elle-même (qui comprend peut-être des procédures d'utilisation…)

L'enquête elle-même est très simple : elle consiste en deux questions posées. Ici, le questionnaire porte sur le logiciel :

1. Notez de 0 à 10 le logiciel
2. Que faudrait-il pour que votre notre soit 10 ?

[15]Ce jeu est disponible gratuitement en ligne : perfectiongame.org. Ce jeu fait partie des core protocols.

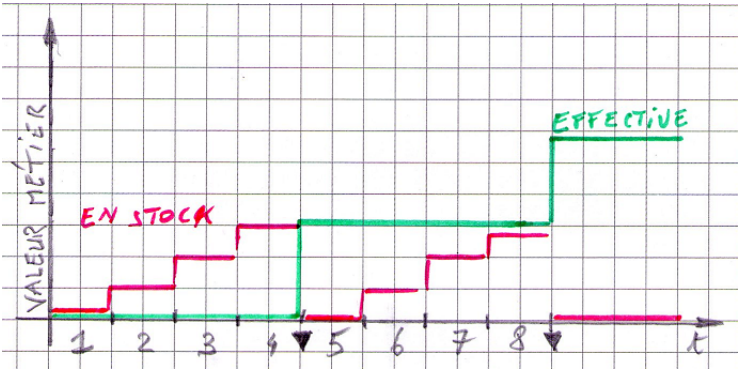
La deuxième question est ouverte. Si le dépouillement des réponses est plus long que dans le cas de questions fermées, le feedback obtenu est certainement beaucoup plus riche et peut provoquer des améliorations sensibles.

3.2.4 Valeur par consentement

Le protocole de gestion par consentement (chapitre Intelligence collective) permet d'obtenir le consentement sur une décision d'échelle de valeur. Brièvement, ce protocole permet d'obtenir le consentement de tous sur une proposition qui est améliorée en fonction des demandes de clarification et des objections que chaque participant peut émettre.

3.2.5 Surveiller la valeur métier

Le Product Owner a tout intérêt à surveiller, *monitorer*, la valeur métier du produit.
La valeur métier d'une fonctionnalité "terminée" (validée par le Product Owner) est nulle tant qu'elle n'est pas mise en exploitation.
Le stock de valeur métier "potentielle"s'accumule tant que les fonctionnalités ne sont pas effectivement mises en exploitation, c'est-à-dire utilisées. La stratégie du Product Owner consiste donc à éviter ce gaspillage qu'est une valeur métier "dormante" dans le stock.
Le graphique suivant trace la valeur métier.



Valeur métier en stock et effective

Pendant quatre itérations, la valeur métier s'accumule dans le "stock". Concrètement, elle reste nulle. Vient une première mise en production/exploitation qui rend effective cette valeur métier jusque-là inutile à l'organisation. Le même phénomène se reproduit lors du développement de la deuxième version (itérations 5 à 8).
Le même cas de figure sans la première mise en exploitation ne ferait qu'empirer le phénomène : la valeur métier en stock s'accumule pendant huit itérations dans ce cas-là.

Lorsque la première version d'un produit constitue l'essentiel des features - par nécessité ou par manque d'agilité - une solution consiste à ajouter dans le plan une deuxième version, quelques semaines après. Cela permet de rectifier si nécessaire, en fonction du feedback concret de la première mise en exploitation.

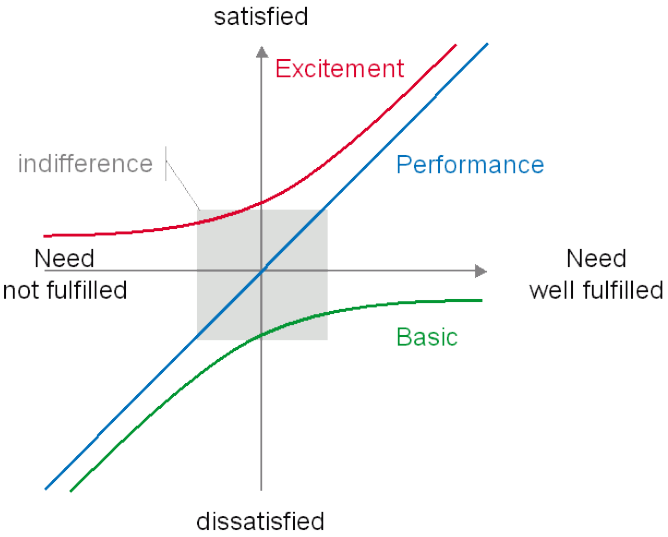
3.2.6 Pour aller plus loin

Lorsque la valeur métier est estimée grâce à la contribution du "vrai Utilisateur", lorsque cette valeur est "monitorée" après une mise en exploitation pour valider l'estimation initiale, alors il est possible d'aller plus loin, si cela s'avère pertinent.

- **Modèle de Kano** : ce modèle[16] distingue plusieurs types de besoins - basique, linéaire, heureuse surprise - et propose un système de questions adressés aux Utilisateurs. Piloter par la valeur métier (l'estimer, vérifier sa réalité après une mise en exploitation, rectifier) reste toutefois plus important que le modèle de Kano ;
- **Estimation absolue** : la pratique "Business Value Poker" établit des estimations *relatives* de valeur métier, ce qui est nécessaire et suffisant pour planifier le développement. "Valeur actuelle nette" est une pratique qui aide à établir la pertinence d'un investissement[17].

Ci-dessous le modèle de Kano[18].

[16] Jean Claude Grosjean en parle sur son site Le modèle de Kano, si précieux....
[17] Voir Valeur Actuelle Nette.
[18] http://fr.wikipedia.org



Modèle de Kano