L'art de la démo

Olivier Ridoux

L'art de la démo

Olivier Ridoux - 2020

Motivations

La **démo** (***demo*** en anglais, pour ***demonstration***) est le pendant empirique de la **démonstration formelle** telle qu'on la pratique en mathématiques. La seconde cherche à convaincre par la raison et vise à l'universalité, tandis que la première le fait par une expérience bien choisie. Et de la même façon qu'il peut y avoir des démonstrations formelles **peu élégantes**, voire **incorrectes**, il peut y avoir des démonstrations empiriques **inadaptées**, voire **contre-productives**.

La démo est le passage obligé de beaucoup de projets, soit pour juger de leur état final, soit pour juger d'étapes intermédiaires. Elle est partie intégrante des **démarches agiles** puisque celles-ci valorisent la production de fonctionnalités sur la production de documents. Elle peut constituer l'élément principal de l'évaluation d'un projet. C'est aussi un passage obligé de nombreux exercices pédagogiques, style travaux pratiques ou projets, durant lesquels des étudiants doivent régulièrement présenter un état d'avancement de leurs travaux.

Une démo doit donc se préparer et se valider, comme un projet. Elle est partie intégrante du projet qu'elle est censée défendre. Et on va voir qu'on peut même se servir de la préparation de la démo pour piloter le projet.

Préparer une démo consiste en scénariser son déroulement de façon à **valoriser le maximum** d'une réalisation en risquant un **minimum d'aléas**. Cela peut aussi consister en un **infléchissement du développement** de façon à avoir de meilleures choses à présenter. Préparer une démo consiste enfin en des **répétitions** et la préparation de **matériels spécifiques**, comme pour toute mise en scène.

L'objectif de ce document est donc de **sécuriser** la communication de ceux qui ont à présenter une démo, de leur présenter un point de vue **pratico-pratique**, et de leur faire se poser les **bonnes questions**.

Comme pour tout exercice de communication on doit d'abord se demander pour qui est faite la démo :

• Pour un utilisateur ? Il sera attaché aux fonctionnalités, et à la facilité d'usage, facilité incluant efficacité. Il sera souvent inquiet des changements qu'une nouvelle application entraîne.

• Pour un expert fonctionnel ? Il voudra comprendre comment votre application se compare par rapport à celles qui existent déjà. Il fera souvent le biais : **Ça existe déjà** ou **L'application X permet aussi de le faire**.

• Pour un expert technique ? Il voudra comprendre comment votre application est réalisée, et quelles conséquences cela a en particulier concernant des propriétés non-fonctionnelles comme la sécurité ou la fiabilité. En 2015, il s'inquiétera de la sécurité, et en 2038 il demandera où est l'IA.

• Pour un financeur ? Il ne se positionnera pas forcément comme un futur utilisateur, et ne comprendra pas grand chose aux détails techniques, mais il voudra savoir quels bénéfices on obtiendrait à utiliser le système en démonstration. Il pourra demander de mutliter un projet pour mieux le comprendre.

Il est très possible qu'un même produit doive être présenté à ces différents publics, et même à plusieurs d'entre eux à la fois. Cela ne fera que compliquer les choses, mais de toute façon tout ira beaucoup mieux en en ayant conscience.

Dans tous les cas, il faudra trouver une mise en scène adaptée : ex. très empathique pour le futur utilisateur, plutôt marketing pour le financier, et assez érudite pour les experts. Noter enfin que même si on ne comprend pas les réponses on est souvent capable d'apprécier qu'elles ont convaincu. Le banquier pourra donc apprécier que des questions de sécurité ont bien été traitées, même si il n'y connaît rien.

David Ogilvy, surnommé le ***Père de la publicité***, disait ***If you find yourself one fine day saying the same things to a bishop and a trapezist, you are done for.***

Souvent, une démo prend la forme d'une mise en scène autour d'une situation **prétexte**. Une des difficultés est de faire que le prétexte soit adapté au public. Mais la plus sournoise des difficultés est sans doute d'éviter que le prétexte ne prenne le pas sur le produit en démonstration, alors que la tentation est grande de faire de la démo un ***show***.

Un proverbe chinois dit **Le doigt du sage montre la Lune, l'idiot voit le doigt.** Transcrit dans le contexte de la démo, cela donne **La démo montre l'application, le public voit la démo**. En d'autres termes, le prétexte choisi pour montrer l'application masque l'application. Peut-être est-elle plus intéressante/séduisante que l'application elle-même ? Mais ici, il ne s'agit pas de qualifier le public d'idiot !

On se rappellera l'application **ELIZA** dans laquelle un nombreux public a vu dans les années 60 un agent conversationnel à l'écoute et bienveillant. Pourtant **Joseph Weisenbaum** l'avait conçue pour démontrer les limites d'une interface homme-machine en langue naturelle. C'est un cas où la créativité du démonstrateur a trahi son objectif. Un autre exemple, vécu, est celui d'un professeur d'informatique qui va présenter à ses collègues physiciens son prochain cours pour leurs étudiants. Ne connaissant pas l'environnement informatique de ses collègues il décide de faire une démonstration sur son PC portable, un des tout premiers qui ait été commercialisé. Résultat, les physiciens se sont passionnés pour le PC portable, et particulièrement de la technologie de sa batterie, et ont oublié l'objet de la rencontre.

Une façon d'éviter ce danger est un scénario en deux temps :

1. Un premier temps présente l'objet de la démo dans une sorte de **paquet neutre** : une application minimaliste qui ne demande pas à réfléchir, une application si simple que le public comprend d'emblée ce qui doit se passer. Et précisément, cette partie de la démo sera convaincante si il ne se passe que ce qui doit se passer. Cette partie est donc très orientée **fonctionnalité**. En terme CAB (**capacités**, **avantages**, **bénéfices**) cette partie présente des **capacités**.
2. Un second temps utilise un prétexte plus théorique, adapté aux attentes du public, pour mettre en scène le produit. En terme CAB, cette partie présente des **bénéfices**. Cette partie pourra démontrer la capacité de traitement du produit sur des données tellement complexes ou nombreuses qu'il est beaucoup plus difficile de prévoir ce qui doit se passer. Mais là, il sera nécessaire de commenter la réponse et d'expliquer son sens, car on peut rarement se contenter de la réponse **42**. Il est aussi important de ne pas seulement laisser les bénéfices apparents, mais de les souligner afin d'éviter tout malentendu.

La difficulté suivante est celle du compromis à trouver entre laisser au public une grande latitude de pilotage de la démo, au risque de laisser voir des anomalies, et un contrôle strict de la démo par le présentateur, au risque de frustrer le public.

Au **prince Potemkine**, ministre de Catherine II, tsarine de Russie, s'est attaché la réputation de décorer les villages misérables que devait traverser la tsarine afin qu'ils donnent une impression de prospérité. Cette réputation semble être immérite, mais l'expression de **village Potemkine** est restée.

Pour une démo, on doit évidemment exclure de montrer un simulacre d'application comme si il était l'application. Cependant, il n'est pas interdit de montrer des simulacres, et c'est même très fréquent, ex. quand on montre une maquette, un plan, ou une visite en réalité virtuelle. Ce qui est interdit est de les présenter comme réels, car alors ce serait des villages Potemkine. Il ne faut pas que le public puisse avoir ce doute. Et pour cela, rien de mieux est de permettre au public de choisir des ingrédients de la démo qui vont en changer le cours. Mais cela risque de déclencher un comportement insuffisamment testé et finalement une anomalie.

En fait, tout va dépendre de l'état réel du produit présenté par une démo, et c'est là un des grands intérêts d'une démo bien faite : montrer l'étendue de ce qui a été réalisé, y compris en robustesse, qualité de service, etc. Si le produit est très défaillant, on pourra présenter qu'une démo extrêmement bien balisée et qui ne laisse la place à aucun imprévu, mais le public sera vite déçu. A contrario, si le produit est vraiment utilisable, on pourra laisser le public intervenir, et celui-ci pourra plus facilement se positionner comme futur utilisateur.

A nouveau, un scénario en deux temps permet de limiter les risques :

1. Dans un premier temps, faire une démo complètement balisée dont le seul objectif est une visite guidée. La démo paquet neutre du premier scénario peut aussi servir à cela. Ici, l'imprévu est exclu car la visite aura dû être répétée et reporduite fidèlement. **Toute anomalie est impardonnable**.
2. Dans un second temps, faire une démo où le public peut intervenir et infléchir la visite. La prise de risque peut devenir importante, mais si le démonstrateur sait à l'avance que quelque chose va mal se passer, il vaut mieux qu'il l'avoue au public plutôt que d'aller exprès dans le mur. Ici, une anomalie peut s'expliquer, mais elle révélera aussi que le produit n'est pas au point.

On entend souvent invoquer **l'effet démo** pour excuser une démo qui se passe mal. Cela fait partie du folklore pessimiste de l'ingénierie : loi de **Murphy**, etc. Il est bon de savoir rire de ses malheurs, mais on ne peut pas invoquer l'effet démo pour une démo qu'on a ratée ! C'est au public d'avoir cette bienveillance, s'il le veut bien.

Si le scénario de la démo est précisé suffisamment à l'avance, réaliser de la meilleure façon possible les fonctionnalités utilisées par le scénario devrait devenir un objectif de développement.

Par exemple, prenons un développement qui a déjà produit les fonctionnalités A, B et C ; B et C étant assez proches l'une de l'autre, et à assez distante de B et C. La distance entre deux fonctionnalité se mesure en l'effort qu'il faudrait fournir pour les solliciter dans le même scénario. Dans cette situation, un scénario autour de B et de C paraît un bon choix, mais pour être cohérent il faudra donner la priorité au développement de ce qui manque pour réunir B et C. Et cela peut même conduire à renoncer à poursuivre le développement de A.

Bien sûr, la scénarisation de la démo ne peut pas être un seul argument. Ex. si A est la priorité n° 1 du client, c'est bien A qu'il faut développer et scénariser pour lui montrer que les choses avancent. Et si c'est l'ensemble A et B qui est prioritaire, il ne faillait sans doute pas développer C. Ces questions de planification doivent être réglées en cohérence avec le projet global, les attentes du client, l'architecture du produit, et les ressources disponibles. Mais réfléchir à la démo suffisamment longtemps à l'avance donne des arguments pour la planification.

L'avantage d'intégrer la démo à ces réflexions est de s'obliger à préciser très concrètement les attendus du projet. Sans cela, il est fréquent de s'accorder sur des objectifs trop abstraits (souvent parce que trop consensuels, comme par exemple, pouvoir se connecter, ou pouvoir passer une commande) pour ensuite avoir un désaccord devant une réalisation sincère de l'objectif.

La **démo** sert à convaincre de la réalisation **d'objectifs**, selon l'adage ***The proof of the cake is in the eating***. Elle est souvent le seul moyen d'y arriver, ex. pour des objectifs non fonctionnels comme **être efficace**, ou des objectifs très mal formalisés comme **être utile**. On ne doit pas la considérer seulement comme un événement social, à côté du développement. Elle en fait partie intégrante, et elle doit être gérée comme tel. Les pratiques **agiles** tendent à rendre cette relation explicite.

La démo et le développement avancent de paire, au point qu'on peut y penser comme à un ***Demo Driven Development*** (DDD). Cette idée est parfois décriée car elle pourrait conduire à des **villages Potemkine**, mais si cela se produit c'est probablement parce que les objectifs de la démo n'avaient pas été assez bien précisés, ou qu'ils aient été purement marketing. Au contraire, réfléchir à la démo très en amont permet de réfléchir aux scénarios attendus comme à ceux dont on ne veut pas. Cela souligne l'obligation de penser à l'évaluation d'un projet sur la base d'un objectif partagé par toutes les parties prenantes. Et faire de l'objectif de développement un objectif de démo permet de le rendre palpable. Évidemment, il aura fallu être sincère dans le passage de l'un à l'autre.

Puisque une démo a pour objectif de **convaincre**, elle est soumise aux aléas de la communication inter-humaine. Elle ne doit pas masquer ce qu'elle est censée exposer, ni laisser penser qu'elle ne marche que pour celui qui présente. Pour cela des scénarios simples permettent de bien séparer la présentation du produit ou du service, de celle de son utilisation dans un cas réaliste, et de bien séparer les moments où le démonstrateur agit pour son compte de ceux où il agit pour le compte d'un utilisateur.

Bibliographie

• « ***The Mother of All Demos*** », Douglas Engelbart (1968).
• « ***The Theory and Practice of Selling the Aga Cooker - Training Manual*** », David Ogilvy (AGA Heat Limited, 1935).
• « ***"Demo or Die" or "Deploy or Die": A Lesson for Education?*** », David Croteau (https://davidrcroteau.net/blog-post/demo-or-die-or-deploy-or-die-a-lesson-for-education/, 2014).
• « **Anthropologie de la démonstration** », Claude Rosental (Revue d'anthropologie des connaissances, 2009) .

Sans se limiter strictement au **jour d'avant**, ni au cas de la préparation d'une démo, il est important de bien gérer le temps qui précède une échéance.

Dans le cas de la préparation d'une démo, il y a au moins deux choses à faire : préparer la démo proprement dite, son scénario, ses supports, ses données, etc., et faire avancer le développement du projet. On vient de le voir, les deux s'entrelacent, la préparation de la démo pouvant exercer une pression sur les objectifs de développement, et l'avancement de celui-ci constituant comme une force d'inertie pour le développement de la démo. C'est pourquoi la préparation de la démo doit être intégrée au développement du projet.

Du côté démo, il est indispensable d'avoir pris le temps de répéter. Répéter ne se limite pas à exécuter la démo sur son terminal en se récitant ce qu'on doit dire. La répétition doit se faire le plus possible dans les conditions de la démo, et c'est seulement comme cela qu'on pourra déceler que les couleurs sont mal choisies, que quelque chose est écrit trop petit, qu'un message bizarre apparait sur la console, que la salle où se passera la démo est derrière un pare-feu qui empêche l'accès à Internet, ou que Sidney n'est pas la capitale de l'Australie. Plusieurs répétitions peuvent être nécessaires.

Il n'est pas interdit d'accompagner la démo d'une présentation plus formelle, qui explique les objectifs, ce qu'on va voir, à quoi s'attendre et pourquoi. Il ne faut pas hésiter à souffler les conclusions à l'auditeur afin d'augmenter ses chances qu'il retienne les bonnes choses.

Une fois les répétitions faites et la démo bien calée, il faut surtout ne plus toucher à rien ! Même avec les meilleures intentions du monde, comme résoudre une anomalie ou « améliorer » un service. Le risque est trop grand de casser quelque chose à ce moment si crucial.

On distingue parfois le ***feature freeze***, pendant lequel on n'ajoute pas de nouvelles fonctionnalités, mais on peut corriger et améliorer les fonctionnalités existantes, et le ***code freeze***, où on ne touche plus du tout au programme. Un système de gestion de versions, ex. **Git**, permet de gérer cette injonction avec souplesse, en figeant une branche pour la démo, ou l'exploitation, et en en démarrant une autre pour le développement.

Un des aspects qui distinguent l'agilité d'autres démarches de développement logiciel est de donner un statut à la démo. Ce statut est celui de moyen d'obtenir un avis du client (le ***product-owner*** ou **PO**) sur le développement en cours. Cela se passe de façon très protocolaire en fin de ***sprint*** pour avoir l'avis du client sur les résultats du sprint, mais peut être utile à tout moment puisque le client **est dans la boucle**.

Le grand principe à l'œuvre ici est que le client n'est pas un spécialiste de l'informatique et l'équipe de développement n'est pas forcément spécialiste du domaine d'activité du PO non plus. Cela a deux conséquences :

1. Selon l'adage que le client est roi, la communication avec le client doit se faire dans ses termes à lui.
2. Même avec la meilleure volonté du monde, le client et l'équipe de développement peuvent tomber d'accord sans avoir vraiment compris la même chose.

En agilité, on suppose que ces deux conséquences sont **irréductibles**. Il n'y a pas de méthode générale qui permette de résoudre à tout les coups ces deux difficultés. Très souvent, on admet que c'est encore le cas même quand le client et l'équipe partagent le même domaine. De ce fait, la démo est un bon moyen de parler au client dans ses termes (conséquence 1), et de lui présenter la compréhension de l'équipe de développement pour avoir son avis (conséquence 2).

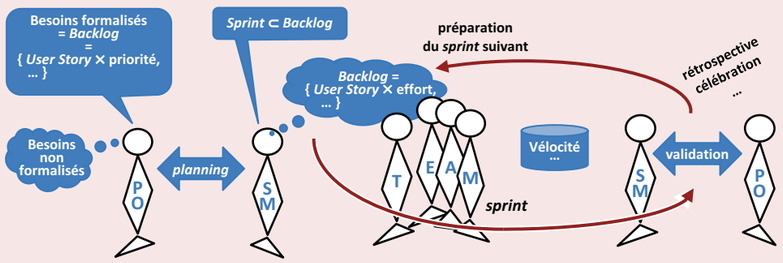
Dans le cadre assez formel de la revue de fin de *sprint*, la scénarisation devrait se limiter à ce qui a été demandé lors de l'expression des ***user stories***. Mais dans le cadre plus exploratoire de profiter du client dans la boucle la démo, elle peut servir à formuler des questions ***What-if*** (Que doit-il se passer si ... ?).

La démo attitude

On l'a vu, la démo est un acte de communication et est donc destinée à convaincre ou à comprendre autrui. On attend donc des démonstrateurs une attitude qui le montre, c.-à-d. une attitude tournée vers autrui. Pourtant, on observe trop souvent un excès de ***cool attitude*** qui à force de distanciation donne une impression de désengagement et de nonchalance. Dans le cadre de relations professionnelles qu'on veut durables, il est important de ne pas confondre **engagement** et **stress**, mais pour que ces relations restent efficaces il est important de ne pas perdre l'engagement en voulant éviter le stress.

Prérequis : agilité et communication - lire, apprendre, comprendre.
Pliage : recto visible (autre face), traits **gris** rentrants, traits **rouges** saillants. Découper selon le trait rouge entre les deux ● de l'autre face, puis achever le pliage.

① Le cycle de vie agile



Le client (***product owner*** ou **PO**) présente ses besoins sous la forme de ***user stories*** (des cas d'usage) agrémentées d'une indication de **priorité**. L'ensemble constitue le ***backlog***. Les *users stories* s'expriment le plus souvent en termes client (quand on fait ceci, on obtient cela) qui peuvent servir de cas de test. Le ***scrum master*** (SM) et son équipe de développement évaluent l'**effort** demandé par chaque ***user story***.

Les priorités s'expriment en valeurs **ordonnables** (1, 2, etc.), mais pas additives. C-à-d. pas dans l'esprit d'exprimer que 2 est deux fois plus prioritaire que 1. L'effort s'exprime en **point d'effort**. Ce sont des valeurs **ordonnables** et **additives** ; 4 représente un effort double de 2. On ne cherche pas à faire une estimation très fine, mais plutôt à dégager des classes. Pour cela on adopte souvent des échelles standardisées qui obligent à écarter les estimations, comme l'échelle de Fibonacci (1, 2, 3, 5, 8, etc.)

Les *user stories*, priorités et points efforts sont les données de base de la planification qui permet de constituer des ***sprints*** de durée fixée à l'avance, ex. 2 semaines, et qui satisfont au mieux le PO (en privilégiant les *user stories* de plus grandes priorités) tout en étant compatible avec la capacité d'effort de l'équipe (Σ effort $\leq$ capacité).

À la fin d'un *sprint*, le PO valide, ou non, les *user stories*, exprime un retour sur les services produits, et peut modifier le *backlog*. La validation se fait essentiellement via des **démos** qui réalisent les cas d'usage attendus. Puis, tout le monde planifie un nouveau *sprint*, et on recommence.

En permanence, la vélocité de l'équipe est mesurée en terme de points d'effort par *sprint*. Sans valeur absolue ces mesures permettent à l'équipe d'affiner son estimation de l'effort demandé par chaque *user story*.

Ce modèle itératif a entre autre la vertu de permettre une amélioration continue, mais assez rapide, via des expériences (ex. l'évaluation de l'effort et la planification) reproduites assez souvent. Plus généralement, l'agilité permet que les erreurs des uns ou des autres (ex. expression des besoins, analyse, programmation, estimation des coûts) soient détectées et corrigées le plus tôt possible.

N'oubliez jamais !

Même dans le cadre d'une activité technique, la communication exige **rigueur** et **correction** dans l'emploi de la langue : orthographe, grammaire, argumentation. Mais aussi dans le choix des circonstances utilisées pour illustrer le propos dans une démonstration (contre-exemple vécu : des informaticiens, spécialistes de la simulation de l'activité du cerveau, qui expliquent **« On s'amuse à mettre une tumeur, et on regarde ce qui se passe ! »**). Choisir des circonstances tragiques ou très sensibles permet sans doute de capter l'intérêt, mais cela suscitera le rejet quand elles sont traitées avec désinvolture. En plus, cela expose à ce que l'auditeur naïf ou très concerné ne voient qu'elles.

① Le diaporama qui aide

Il est souvent utile d'introduire et de conclure une démo par un **diaporama** (ne jamais dire un **power-point** ! Ne pas confondre l'objet avec l'outil qui l'a fabriqué.). L'objectif est d'aider le présentateur à ne rien oublier de ce qu'il a à dire, et l'auditeur à retenir ce qu'il faut retenir.

Surtout ne pas suivre les recommandations stéréotypées du genre **pas de phrase, seulement des mots clés** ou **pas plus de 6 points par diapo, et pas plus de 6 mots par point**. S'en tenir aux objectifs :

- **Aider le présentateur** : un bon moyen de perdre son auditoire est de se rendre compte à la diapo 15 qu'on a oublié de dire quelque chose à la diapo 5. Pour éviter cela, la scénarisation permet d'identifier les points importants, et les répétitions permettent d'identifier les points qu'on oublie tout le temps. Il vaut mieux un diaporama un peu verbeux mais qui aide, qu'un diaporama qui entretient la confusion.
- **Aider l'auditeur** : il a ses faiblesses, ex. des moments d'inattention, des lacunes, des présupposés erronés. Les hypothèses et les conclusions de la présentation doivent être affichées clairement. Dans tous les cas, c'est la faute du présentateur si l'auditeur ne retient pas la bonne chose.
- **Aider les deux** : il est vain de présenter sans support écrit des noms propres (personnes, lieu, société, produit, marque, ...), sauf si ils sont déjà très connus. Pareillement, pour des quantités, dates, etc. Tout cela doit être écrit si on veut que ce soit retenu. Et si il n'est pas important que ce soit retenu, pourquoi en parler ? Sans support écrit, une phrase comme **En 19xx, M. X créait la société S, dont le siège est à T, qui a maintenant Y salariés dans Z pays, pour un chiffre d'affaire de C** est du gâchis de temps, soit que ces données ne seront pas mémorisées, soit qu'elles sont déjà connues.

② Le diaporama qui nuit

Un excellent moyen pour un diaporama de nuire à un orateur et de ne pas dire la même chose que lui, ou pire de le contredire.

Nous sommes ainsi faits que généralement le canal écrit nous demande plus d'attention que le canal oral, mais est plus rapide. En conséquence, le diaporama qui nuit le fait parce qu'il prend le dessus sur l'orateur. Cela peut arriver de plusieurs façons :

- Le diaporama présente de l'information qui n'est pas oralisée. C'est le cas typique de diaporamas trop fournis et que l'orateur ne fait que survoler. Immanquablement, l'auditeur va se lancer dans une course de vitesse avec l'orateur pour déchiffrer le diaporama plus vite que l'orateur ne le survole.
- Le diaporama présente la même information que celle qui est oralisée, mais il la présente autrement. Cela crée un conflit cognitif que l'auditeur résoudra en ignorant l'orateur.
- Le diaporama parle d'autre chose que ce que dit l'orateur. Ou dit le contraire ! Cela semble absurde, mais cela arrive.
- Le diaporama présente exactement la même chose que l'orateur ; en fait, c'est comme si l'orateur le lisait. Dans ce cas, le mieux qu'ait à faire l'auditeur est de le lire aussi.

Pour que le diaporama aide il faut que la relation entre ce qui est dit et ce qui est écrit soit extrêmement simple. Par exemple, que soient écrits les points qui sont développés à l'oral, ou que soient écrites les hypothèses, définitions et conclusions à retenir. Et tout ce qui par nature passe mal à l'oral : orthographe, chiffres, noms propres, formules, etc.

③ Le vu et le lu

Lire demande un effort d'attention que ne demande pas le simple fait de voir. Cela explique des recommandations/oukases qui interdisent d'utiliser des phrases dans un diaporama, ou même limitent le nombre de mots par page. Ce sont des recommandations qui deviennent absurdes quand elles font fi du contenu réel de la communication. En effet, il est des contenus qui doivent être lus littéralement : poésie, articles de loi, citations, mais aussi formules, spécifications... Ce sont des données de certaines communications.

D'autres données doivent souvent être écrites : noms propres, de personnes, de localités, de produits, quantités, ..., car sans cela elles se transforment en bouillie sonore, et à condition évidemment que ces données servent dans la suite de la présentation.

Les textes conçus pour être **vus** et ceux pour être **lus** sont souvent entourés de modalités typographiques différentes. Par exemple, les guillemets signalent une citation verbatim, donc un texte pour être lu. Les polices avec empattements (**serif**, en anglais) sont souvent le signe d'un texte pour être lu, alors que les polices sans (**sans serif**) sont celui d'un texte pour être vu ; comparer par exemple la typographie des affiches, pour être vu, et celle des imprimés classiques, pour être lu. Même si l'impression moderne utilise de plus en plus souvent des polices sans serif (par exemple, ***calibri*** pour ce document) pour des textes à lire.

L'impact des empattements sur la lisibilité est très débattu, des études se prétendant scientifiques démontrant une chose ou son contraire. Il faut d'abord se souvenir du rôle du support (ex. pierre, bois, papier, écran), du procédé de transfert (ex. gravure, peinture, gravure, rayon cathodique), et de la qualité de l'ensemble. Des arguments qui tenaient pour des écrans de mauvaise qualité peuvent ne plus tenir pour des écrans modernes, et encore moins pour du papier électronique. Mais une chose est certaine, avec et sans empattement constitue un contraste visuel important qui permet de distinguer facilement le **vu** et le **lu**.

Une autre chose est certaine, le **vu** et le **lu** sont deux modalités de présentation tout aussi légitime l'une que l'autre. Il faut seulement les utiliser à bon escient.

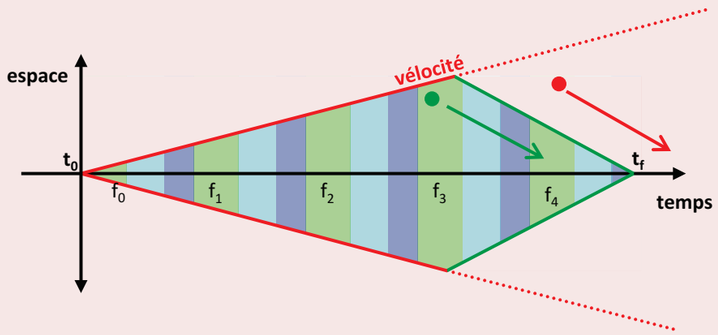
② La démo et l'espace-temps de l'agilité

À partir du moment où on fixe une limite de vitesse pour une progression, on sait qu'il n'est pas possible d'avancer d'une quantité arbitrairement grande en un temps donné. C'est un principe qui vient de la physique, où il a été considérablement renouvelé par la relativité de Einstein. On emploie souvent ce principe sous la forme d'un **cône d'accessibilité** qui s'élargit vers le futur et délimite l'espace qu'on peut atteindre en un temps donné. Par symétrie, on peut concevoir un cône d'accessibilité vers le passé. Il délimite les points de l'espace d'où on peut être venu en un temps donné.

Ce principe s'applique aussi à la gestion du temps dans un projet de développement logiciel. Dans ce cas, l'espace n'est pas celui de la physique, mais celui des fonctionnalités à réaliser. Le plus dur est de connaître la vitesse de développement. Et c'est même une des choses les plus difficiles qui soit à faire, alors que avoir une idée de la vitesse de développement est essentiel pour estimer le coût d'un projet. Les approches agiles à la **Scrum** propose un protocole d'estimation de la vitesse tout au long de la vie d'une équipe.

L'idée est de confronter les estimations de coût en **points d'effort** faites lors de la planification d'un ***sprint***, avec la quantité de fonctionnalités, les ***user stories***, réalisées à la fin du sprint, et mesurée comme la somme des points d'effort des *user stories* validées. On en déduit la **vélocité** de l'équipe, mesurée en point d'effort par durée de sprint. La vélocité peut être modérée par la **disponibilité** de chacun sur le projet ou pour un *sprint* (ex. pour gérer les périodes de vacances). L'objectif est d'obtenir une estimation toujours meilleure de la vélocité.

Une fois cela mis en place on peut imaginer le diagramme espace-temps d'un projet de développement agile. Il alterne des *sprints* qui comprennent chacun un peu d'**analyse**, de **développement** et d'**intégration et validation**. Le cône d'accessibilité s'élargit avec le temps (contour **rouge**, ouvert à droite) et permet d'atteindre de plus en plus de fonctionnalités. Un autre cône, tourné vers le passé (contour **vert**, ouvert à gauche), modélise ce qui peut être fait avant la fin du sprint (t_f). On voit des fonctionnalités qui valent la peine d'être commencée (point **vert**) car il est possible de les terminer avant la fin du *sprint*, et d'autres (point **rouge**) qu'on pourrait commencer, mais pas terminer.



Les préparatifs de la démo qui termine souvent chaque sprint doivent donc être inscrits dans ce diagramme, que ce soit en **analyse**, **développement** ou **intégration et validation**. Par ailleurs, d'autres démos peuvent avoir lieu tout le long du sprint pour s'expliquer, se faire comprendre, ou convaincre. Ces démos impromptues seront d'autant plus faciles à faire que l'infrastructure de démo aura été intégrée au développement.