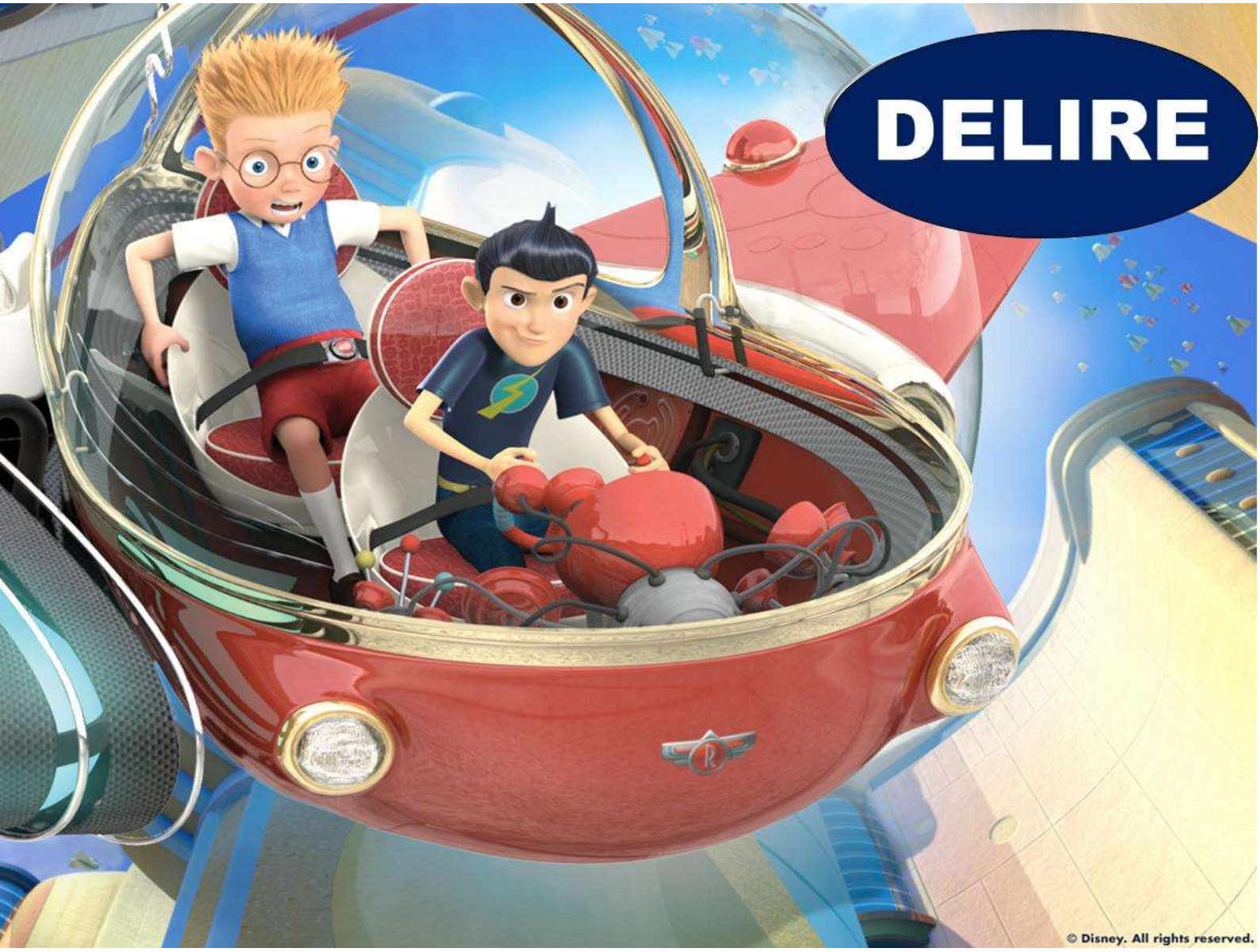

Le projet **DELIRE**

Développement par **E**quipe

de **L**ivrables **I**nformatiques

et **R**éalisation **E**ncadrée

Que contiennent nos livrables

Introduction
Pour beaucoup d'entre vous, un projet informatique c'est essentiellement se jeter sur son clavier pour pisser du code, secouer le résultat jusqu'à ce qu'il tombe en marche, et considérer alors le travail comme terminé.
Ceci fonctionne bien lorsqu'il s'agit d'un TP sur un sujet technique donné. Malheureusement, le projet DELIRE a une autre ambition. L'objectif est de vous mettre en situation de développement de projet dans un environnement industriel.

Pour mémoire, voici la définition normalisée d'un projet : action spécifique, nouvelle, de durée limitée, qui structure méthodiquement et progressivement une réalité à venir.

La démarche va constituer, à partir d'un document initial fourni par le client, le Cahier des Charges, à progresser dans la compréhension du besoin, à formuler la réponse qui nous semble pertinent, à définir comment on va le faire, quand on va le faire, pour quel coût, comment on va s'organiser, quels sont les objectifs de qualité du produit visés… au travers de documents écrits qui s'appellent les livrables

Dans le résumé « Comprendre la gestion de projet complexe pour bien démarrer le projet », nous avions posé que le projet DELIRE se décomposait en quatre phases : Spécification, Structuration, Réalisation et Convergence.
Ce document présente des éléments de base pour rédiger les livrables de chacune des quatre phases.

Les livrables de la phase de Spécification
Les buts de la phase de Spécification sont de

- Comprendre les besoins et les contraintes (techniques, de délai et de coût) du client
- Proposer au client une réponse qui vous permette de remporter le marché en gagnant de l'argent
- Définir les points clés de votre solution (incluant les facteurs différentiateurs) vos choix techniques, le séquençage des versions successives mises à disposition du client
- Définir l'enchainement des tâches en fonction de la typologie de l'utilisateur
- Pouvoir présenter au client et à l'équipe de développement l'aspect ergonomique qu'aura votre produit.
- Avoir une bonne idée de la façon dont vous aller partager le travail au sein de l'équipe de développement

Le Cahier des Charges

- Il est rédigé par le MOA, qui représente les intérêts du client
- Il exprime
    - les besoins à satisfaire
    - Le délai proposé
    - L'argent qui est mis sur la table pour réaliser le produit
    - Eventuellement des contraintes techniques : version de Java à supporter sur le serveur, nécessité de pouvoir tourner sur un IPhone 6 avec IOS 10…

Attention, le Cahier des Charges rédigé par Denis Devictor représente le produit idéal à terminaison, mais il est extrêmement ambitieux. Soyez raisonnable dans vos demandes.

Le Business plan MOA

- Il est rédigé par le MOA
- Il identifie l'argent qui sera gagné chaque année par l'entreprise lorsque le produit sera déployé
- On peut considérer que 5% de la somme gagné durant les 3 premières années d'utilisation du produit est la somme que le client est prêt à mettre sur la table pour réaliser le produit.
- Exemple :
    - 20,000 personnes de l'AP-HP utiliseront le produit
    - Elles économiseront 2 heures par an
    - Le cout d'une heure de travail (salaire plus charges) est en moyenne de 30 €
    - Soit un bénéfice de 20,000 * 2 * 30 € = 1,200,000 € par an soit au total 3,600,000 € sur 3 ans
- La somme que le client est prêt à mettre sur la table est 3,600,000 € * 5% = **180,000 €**

Les Spécifications Fonctionnelles Générales

- Elles sont rédigées par le Designer, en collaboration avec l'Architecte et le MOE
- Les Spécifications Fonctionnelles Générales sont la réponse de l'équipe projet au Cahier des Charges.
- Elles définissent
    - Les fonctionnalités principales du produit
    - Les facteurs différentiateurs de votre produit par rapport à la concurrence : support du multi langages, capacité à pouvoir travailler en étant déconnecté de la base de données par exemple.
    - Certains critères de qualité important pour le client : fiabilité, performance de la fonctionnalité principale (moins d'une seconde pour charger un dossier médical en étant connecté sur le LAN par exemple),
    - Les dates de remise des différentes versions du produit
    - Le cout à payer pour chacune des versions et potentiellement par utilisateur
    - Les contraintes techniques pour le client : environnement hard de déploiement (une machine avec N giga de mémoire et N téra de disque pour 1000 utilisateurs), produits prérequisites (ORACLE par exemple), mécanisme de maintenance….
- C'est avec ce document que vous allez remporter l'appel d'offres.
    - Il n'est pas nécessaire que vos Spécifications Fonctionnelles Générales répondent à l'ensemble des besoins exprimés par le client dans son Cahier des Charges.
    - Il faut simplement qu'elles soient meilleures que celles proposées par la concurrence.
- Ne soyez pas trop précis dans votre description
    - D'accord pour : le médecin peut rajouter un diagnostic au dossier médical

- Pas d'accord pour : le médecin choisit un dossier en cliquant sur l'icône [DOSSIER] et en saisissant les champs de sélection, puis il sélectionne l'icône [EDITION]…

Dans un vrai projet industriel, les Spécifications Fonctionnelles Générales sont un document qui engage le fournisseur en termes de Cout, Délai et Qualité

Le Business Plan MOE

- Il est rédigé par le MOE, pour valider que réaliser le projet pet être bénéficiaire, ou à tout le moins équilibré.
- Voici la démarche à suivre
  - En s'appuyant sur l'expérience de l'entreprise, il identifie le nombre d'heures qui lui semble nécessaires pour réaliser le produit décrit dans les Spécifications Fonctionnelles Générales
  - Il multiplie le nombre d'heures estimé par le cout horaire d'un ingénieur : partez sur une hypothèse de 50 € de l'heure (ce n'est pas le salaire net, qui est de l'ordre de 12 à 15 €, mais cela inclut également les charges salariales et patronale, la location des bureaux, eau gaz électricité téléphone, impôts locaux, achat des ordinateurs, la secrétaire, les déplacements……)
  - Rajoutez 20% de dérapage
  - Rajoutez 10% de bénéfice.
- Cela représente la somme que vous souhaitez demander au client pour faire le projet.
- Exemple
  - Vous identifiez 3 versions successives pour le produit de 900 heures chacune
  - 900 * 50 = 45,000
  - 45000 * 1.2 = 54,000
  - 54,000 * 1.1 = 59,400
- Demander 60,000 € par version est donc une bonne base de négociation. Miracle, vous tombez sur la somme de **180,000 €.**
- C'est ce chiffre qui apparaitra dans les Spécifications Fonctionnelles Générales
- Attention : le Business Plan MOE est le seul document dans lequel l'équipe projet parle d'€. Ensuite, vous ne parlerez plus que des 900 heures nécessaires pour réaliser la première version du produit.

Remarque : il est inutile que votre Business Plan MOE pour DELIRE arrive à la conclusion que le projet n'est pas rentable et que dans ce cadre vous renoncez à soumettre vos Spécifications Fonctionnelles Générales au client

L'Architecture Fonctionnelle

- Elle est rédigée par le Designer, en collaboration avec le Responsable Qualité et l'Architecte
- Elle définit
  - l'arborescence des menus
  - la finalité de chaque commande, aux nœuds terminaux de l'arborescence des menus
  - les commandes permanentes : déconnexion, undo, changement de langue, help on ligne, accès à la documentation…

- Elle est pilotée par
    - Le respect d'un standard de présentation et de dialogue
        - La fenêtre de dialogue de sélection d'un dossier médical est la même quelle que soit la commande, et quelle que soit la typologie de l'utilisateur.
        - La meilleure façon de respecter cette consigne est de faire appel systématiquement au même composant logiciel
    - La convivialité de navigation dans les menus, de sélection ou de saisie des données dans un panel
    - Lé réduction du nombre d'interactions
    - La facilité du travail : mieux vaut saisir dans une liste déroulante que d'entrer une chaine de caractère.
    - La simplicité, la simplicité et la simplicité

Le Designer a également la responsabilité de rédiger les scénarios fonctionnels d'utilisation du produit

- Il y a à minima un scénario fonctionnel par typologie d'utilisateur : administrateur, secrétaire, médecin, infirmier…
- Ces scénarios, exécutés à la main vont permettre :
    - De vérifier que toutes les commandes définies dans l'Architecture Fonctionnelle sont utiles
    - De vérifier qu'il ne manque pas de commande
    - De valider la Maquette d'I.H.M.
    - De valider les objectifs de convivialité définis par le Responsable Qualité
    - De valider l'Architecture Logique
- Un scénario a un rôle un peu particulier : c'est le scénario de la démonstration finale.

<u>La Maquette d'I.H.M.</u>

- Elle est conçue par le Designer
- Vous vous adressez à des non informaticiens qui vont chercher les bonnes raisons pour ne pas utiliser votre produit.
- Dans une première approche, dessinez votre I.H.M. à la main
- N'hésitez pas à vous inspirer d'applications existantes et que vous aimez
- Choisissez des icônes standard dans beaucoup de produits
- Pensez à maximiser la surface utile : il n'est pas normal que le logo de l'AP-HP, les menus ou les commandes permanentes monopolisent les 3 quarts de l'écran
- Pensez également que votre produit devra tourner à terme sur une tablette ou un Smartphone
- Utilisez ce dessin de l'I.H.M. pour le faire valider par des personnes totalement extérieures au projet : vos parents, vos amis.
- Et prenez très au sérieux les remarques qu'ils vous font : un I.H.M. mal gaulé peut être à l'origine du rejet d'un produit par ses utilisateurs. .

L'Architecture Logique

- Elle est rédigée par l'architecte
- L'architecture Logique a plusieurs objectifs
    - Définir des composants logiciels (fortement cohérents, faiblement couplés) pertinents
    - Pouvoir distribuer le travail de codage durant la phase de réalisation
    - Permettre les évolutions futures du produit
    - Faciliter sa maintenabilité
    - Supporter le nombre d'utilisateurs nécessaire (exemple, dans notre cas, 20,000 utilisateurs)
    - Mettre en place les mécanismes de sécurité nécessaires, et potentiellement le licensing
    - Définir certains patterns : High Availability, Backup/Restore, réplication, Import/Export, support du multi-langages…
- Un composant logiciel est défini dans l'Architecture Logique par
    - Sa finalité
    - La liste de ses principales fonctions
    - La définition exhaustive et précise de ses APIs (la signature des services qui peuvent être appelés)
    - La liste des autres composants logiciels sur lesquels il a visibilité : en d'autres termes, dont il peut appeler les services selon les APIs dudit composant
- Par contre, la façon dont est agencé l'intérieur du composant ne fait pas partie de l'Architecture Logique, mais des Spécifications Techniques Détaillées, qui sont de la responsabilité du développeur du composant logiciel, en phase de Structuration.

Il y a plusieurs types de composants

- Des composants de présentation
    - Dialogue, saisie des interactions, affichage des panels
    - Ils sont en général structurés par les typologies des utilisateurs identifiées dans l'Architecture Fonctionnelle
    - Néanmoins certains composants génériques (exemple recherche d'un dossier médical) sont partagés entre les différents utilisateurs
- Des composants Business Unit
    - Ils sont structurés par la typologie de l'utilisateur (par exemple médecin, infirmier, secrétaire…) et potentiellement son savoir-faire (neurologue, cardiologue, cancérologue…)
    - Exemples
        - Définition d'un diagnostic par le médecin
        - Validation de l'exécution d'une prescription médicale par l'infirmier
        - Saisie et validation des données sociodémographiques du patient par la secrétaire.
- Des composants de gestion de données
- Ils sont structurés par le modèle de données

Le modèle de données fait partie intégrante de l'Architecture Logique.

- Les grands ensembles que vous devez y trouver à minima sont
    - La structure arborescente de l'hôpital : pôle, services, unités hospitalière, unité de soin…
    - Les informations relatives à un utilisateur : nom prénom, identifiant et password crypté, typologie, spécialité éventuelle, nœud de l'arborescence de l'hôpital auquel il est rattaché…
    - Dossier médical du patient : données sociodémographiques, identifiant (numéro de sécurité sociale si la personne est française par exemple), nœud de l'arborescence de l'hôpital auquel il est rattaché dans le cadre de son hospitalisation, diagnostics et ordonnances, suivi des ordonnances, résultat d'examens…
- La question que vous aurez à vous poser est la suivante : le modèle de données est-il décrit dans un dictionnaire de données externe, par exemple en XML ?
    - C'est extrêmement puissant comme mécanisme : génération des structures de données dans la base de données, gestion des objets en mémoire, capacité à générer automatiquement des panels de présentation, facilité d'évolution des structures de données…
    - Cela vous permet de faire facilement évoluer votre structure de données durant le projet DELIRE
    - Cela vous permettra à terminaison d'échanger vos dossiers médicaux avec des hôpitaux hors AP-HP, y compris des hôpitaux étrangers
    - Cela vous permet d'avoir des composants logiciels génériques, que vous pourrez utiliser dans d'autres projets sans aucun rapport avec DELIRE
    - En termes de défi informatique c'est passionnant
- Par contre, rançon du succès, c'est plus compliqué à mettre au point

La sécurité est une problématique importante, en particulier les problèmes de respect du secret médical et de la traçabilité des actes médicaux
La sécurité est une problématique protéiforme

- Certains aspects se gèrent de façon logicielle, par exemple la connexion à l'application
- D'autres aspects se gèrent plus facilement par une bonne construction du modèle de données.
    - Ainsi, une fois un document stocké en base, il est noté Released.
    - Il n'est plus possible de le modifier ou de le supprimer :
    - Cela vous permet de gérer la traçabilité des actes médicaux (pas de possibilité d'effacer un diagnostic médical, ce qui permettra d'avoir des informations si un éventuel procès pour complication doit avoir lieu 10 ou 20 ans plus tard.
    - Par contre, vous pouvez avoir un état In Work d'un document si par exemple le médecin n'a pas fini son diagnostic et est appelé pour une urgence : en ce cas, il peut le modifier dans les jours qui viennent, mais personne d'autre ne peut y accéder.
- Un médecin ne peut travailler sur un dossier médical que si le patient est rattaché au même nœud de l'arborescence de l'hôpital, ou à un nœud en dessous dans l'arborescence de l'hôpital

L'identification du personnel de l'hôpital.
Le médecin qui fait un diagnostic, ou l'infirmier qui valide l'administration d'une posologie, s'engage.
L'usurpation d'identité est donc dramatique
Même l'administrateur qui va créer le compte du nouvel arrivant ne doit pas connaitre le password du médecin, ou de l'infirmier
Quand l'administrateur crée le compte (emile.durand@ap-hp.fr), cela crée automatiquement la boite mail du nouvel arrivant, qui y reçoit un mail pour activer son compte
La première interaction est de redéfinir et de confirmer son nouveau password, qui est crypté par un algorithme : c'est le cryptage de ce password qui est stocké dans la base de données.
Lors d'une prochaine connexion, on comparera le cryptage du password tapé par l'utilisateur avec cette information stocké en base.

Une question que vous avez à vous poser est de savoir s'il est possible de travailler en étant déconnecté de données

- Par exemple un médecin souhaite finir chez lui un certain nombre de diagnostics.
- Il est sans doute pertinent de prévoir un mécanisme d'export import d'un diagnostic In Work

Bien entendu, dans cette configuration, le médecin ne pourra pas consulter l'historique du patient.

Cet aspect du problème vous conduira à formaliser la structure d'un document médical :

- Il est sans doute pertinent que, lors du stockage en base en état Released d'un document, on puisse en extraire des informations pertinentes pour pouvoir à terme faire des requêtes en base.
- Exemple : liste de tous les patient ayant eu la maladie de Charcot depuis 20 ans.
- Si vous acceptez que le médecin puisse travailler en étant déconnecté, alors se pose la question de l'outil d'édition du diagnostic :
    - Est-ce une sous partie de l'application qui est installés sur le PC du médecin ?
    - Ou bien le diagnostic est un document XML, dont la structure est documentée dans une XSD, et il faut un simple éditeur XML pour le mettre à jour ?

Une autre question que vous aurez à vous poser :

- Les développements que vous aurez à prévoir pour les versions 2 et 3 du produit seront-ils faciles à intégrer, et n'est-il pas nécessaire de prévoir dès la première version des fonctionnalités qui ne seront pas valorisées dans cette première version mais qui permettront de faciliter les futures migrations ?
- Ce peut être le cas par exemple des structures de données qui peuvent comporter des champs non encore utilisés et qui sont valués avec une valeur par défaut, pour éviter d'avoir a posteriori à faire une migration de la base de données.

Le Planning et le Budget

- Il est rédigé par le responsable Planning
- Dans les Business Plans, nous avions parlé €. Dans la suite du projet nous ne parlerons plus qu'heures de travail.
- Avoir un planning et un budget est important dès le départ du projet pour fixer des échéances. Très tôt, il est nécessaire de poser un certain nombre de jalons. La méthode est dite du « doigt mouillé », on fait des estimations à la louche
- Dans cette première version, on va se contenter de définir le planning et le budget des phases de Spécification et de Structuration. C'est-à-dire, la rédaction du Cahier des Charges, , du Business Plan MOA, du Business Plan MOE, de l'Architecture Fonctionnelle, de l'architecture Logique, de la Maquette d'I.H.M., du Plan Qualité, du Plan Management, du Plan de Gestion des Risques, du Plan de Tests, des Spécifications Techniques Détaillées et de de l'Organigramme des Tâches
- Ce planning et ce budget seront ensuite régulièrement mis à jour en fonction de l'avancée du projet. La seule contrainte à respecter est le fait que l'ensemble des livrables des phases de Spécification et de Structuration doivent être remis lors de la soutenance intermédiaire qui aura lieu durant la dernière semaine avant les vacances de Noël.
- Mon conseil concernant le planning
    - Le Cahier des Charges et les Spécifications Fonctionnelles Générales au bout de 2 semaines (T1)
    - Une première version de l'Architecture Fonctionnelle et de l'Architecture Logique au bout de 4 semaines (T2)
    - Une version plus ou moins définitive de l'Architecture Fonctionnelle et de l'Architecture Logique, segmentées par versions, la maquette d'I.H.M. et les deux Business Plans au bout de 6 semaines (T3)
    - Les Plan Qualité, Plan management et Plan de Gestion des Risques peuvent démarrer assez tôt dans le projet
    - Le Plan de test doit attendre la définition claire de la première version du produit (T3)
    - Les Spécifications Techniques Détaillées ne peuvent démarrer qu'après une Architecture Logique globalement figée (T3)
    - L'organigramme des tâches nécessite les Spécifications Techniques Détaillées, ainsi que les Plan Qualité, Plan management et Plan de Gestion des Risques
    - L'élaboration précise du Planning et du Budget des phases de Réalisation et de Convergence, qui devront être disponibles en fin de phase de Structuration, nécessite la disponibilité de l'Organigramme des Tâches.
    - En toute logique une fois le Planning et le Budget des phases de Réalisation et de Convergence établis, il faudrait reboucler sur le budget défini dans le Business Plan MOE (qui est traduit, je le rappelle, en heures de travail), et potentiellement réduire la voilure si il y a une trop grande différence entre le budget visé et le budget obtenu. Nous n'en aurons pas la capacité
- Mon conseil concernant le budget
    - Le Cahier des Charges ; quelques heures
    - Les Spécifications Fonctionnelles Générales ; quelques heures

- Une première version de l'Architecture Fonctionnelle : quelques heures
- Une première version de l'Architecture Logique : comptez facilement 20 heures
- Une version plus ou moins définitive de l'Architecture Fonctionnelle et de l'Architecture Logique, segmentées par versions, la maquette d'I.H.M. et les deux Business Plans) : comptez facilement 30 à 50 heures
- Le Plan Qualité; quelques heures
- Le Plan management : comptez facilement 10 heures
- Le Plan de Gestion des Risques ; quelques heures
- Le Plan de test : quelques heures
- Les Spécifications Techniques détaillées : comptez de 5 à 10 heures par membres de l'équipe
- L'Organigramme des tâches : comptez facilement 10 heures
- L'élaboration précise du Planning et du Budget des phases de Réalisation et de Convergence : comptez facilement 10 heures

Les livrables de la phase de Structuration étape 1

Les buts de la phase de Structuration étape 1 sont ::

- D'avoir identifié les risques potentiels qui menacent le projet et avoir défini des plans de levée
- D'avoir identifié la façon dont l'équipe travaille en collaboration, en particulier durant les périodes de stress
- D'avoir identifié les tests à accomplir pour amener le produit à un niveau de qualité qui permette de le déployer en environnement industriel
- D'avoir identifié l'ensemble des tâches à accomplir en phases de Réalisation et de Convergence

<u>Le Plan Qualité</u>

- Il est rédigé par le responsable Qualité
- Il définit très tôt dans le projet les objectifs qualité du produit, qui pourront être partagés avec le client
  - Convivialité : par exemple, le nombre maxi d'interactions pour une commande de base, le rapport sélection / entrée clavier…
  - Performance : par exemple 1 seconde pour afficher un dossier patient en local (bref, sans latence réseau)
  - Fiabilité
    - C'est assez compliqué : d'habitude on donne le MTBF (Main Time Between Failure) mais dans la réalité, pour garantir moins d'un carton pour 100 heures, il faudrait faire au moins 500 heures de tests.
    - Voici ce que je vous propose : tous les scénarios fonctionnels doivent avoir été exécutés au moins une fois sans carton, et le scénario de la démonstration finale doit avoir été exécuté 3 fois de suite sans carton.
  - Maintenabilité : une documentation de maintenance est disponible
- Il définit un certain nombre de standards que doit suivre l'équipe de développement : standard des livrables (à formaliser rapidement), standard

des présentations, standard des Spécifications Techniques Détaillées, standard des fiches de livraison de composant logiciel, standard de coding…,

Le Plan qualité devra également définir des objectifs de performance pour les composants logiciels, en collaboration avec l'Architecte.

- Si, par exemple, vous avez annoncé 1 seconde pour afficher un dossier patient en local
- Et si l'affichage d'un dossier mobilise 3 composants : REQUETE - STRUCTURATION – AFFICHAGE
- Vous pouvez par exemple tartiner la seconde entre les 3 composants : 0.5 seconde pour REQUETE - 0.3 seconde pour STRUCTURATION – 0.2 seconde pour AFFICHAGE
- Chacun des 3 composants devra vérifier durant ses Unit Tests qu'il respecte son objectif de performance ainsi défini.

<u>Le Plan de Gestion des Risques</u>

- Il est rédigé par le responsable Gestion des Risques
- Dans un projet, on segmente souvent les risques en 4 catégories
  1. Economiques : évaluer et tenir les prévisions budgétaires
  2. Organisationnels : équilibrer délais et ressources, actualiser en permanence les estimations de charges
  3. Fonctionnels : travailler en commun, valider progressivement
  4. Technologiques : maîtriser la nouveauté
- L'objectif de la gestion des risques est de réduire, par des méthodes préventives, opérationnelles ou correctives, les différents risques identifiés et retenus dans chacune des 4 catégories.
- La méthode consiste à :
  - Identifier les risques du projet
  - Les classer par criticité (probabilité * impact) décroissante
  - Pour ceux que l'on choisit de traiter, définition d'un plan de levée préventif (on fait des actions pour réduire la probabilité et/ou l'impact) ou correctif (on ne corrigera e problème que s'il se produit)

Ceci étant posé, au vu de mon expérience du projet DELIRE, 4 risques principaux vous guettent

1. Vous avez choisi dans votre Architecture Logique une technologie que vous ne maîtrisez pas
   - Première solution : faire un prototype durant la phase de Structuration
   - Seconde solution : avoir une technologie de backup
2. Une personne quitte le projet physiquement ou intellectuellement
   - Pour votre gouverne, en 2016/2017, un groupe initialement de 7 personnes s'est retrouvé au final à 4 personnes : autant dire que finir le projet était juste impossible, du fait que le problème n'avait pas été anticipé.
   - J'ai vu également des étudiants avoir des accidents de ski un peu violents, qui les ont immobilisé plusieurs semaines
   - Mais vous pouvez avoir également la personne qui ne répond plus ni aux mails ni aux coups de téléphone et qui ne communique plus rien à ses collègues

- Dans tous les cas, la plus grosse difficulté est de perdre du savoir-faire critique sur le projet : la solution est de toujours doublonner les savoir-faire.

3. La reprise de janvier s'avère compliquée
   - Ne rêvez pas, vous aurez nécessairement le problème.
     - Si le chef de projet est capable de remettre l'équipe en ordre de bataille en moins de 15 jours, cela sera manageable
     - Si par contre, rien ne se passe pendant 4 ou 6 semaines, votre projet est en grand danger.
   - Et dans ce domaine, il n'y a pas de recette miracle : il faut se remettre au travail. C'est la responsabilité du chef de projet. Souvent plus facile à dire qu'à faire.
   - Petite remarque : sur un gros projet comme le Boeing B787, 1 semaine de retard c'est juste… 20,000,000 $ de perdu.
4. Votre projet est trop ambitieux
   - C'est normal : l'esprit humain est par nature optimiste
   - C'est normal, c'est votre premier projet en équipe, et vous ne soupçonnez pas encore les pièges inhérents à ce mode de fonctionnement
   - 1ère solution : décomposez votre projet en 3 versions successives, dont vous en réaliserez que la première version
   - 2nde solution : même dans la 1èfe version, définissez des fonctionnalités qui ne seront développées que si tout se passe bien
   - 3ème solution, de loin la plus payante : soyez modeste dans vos ambitions

<u>Le Plan Management</u>

- Il est rédigé par le chef de projet
- Une des premières tâches du chef de projet est de constituer son équipe.
  - Rappelez-vous que le projet DELIRE est là pour vous apprendre la gestion de projet. Mettre quelqu'un d'inexpérimenté n'est donc pas nécessairement un mauvais choix.
  - Le travail de Designer et le travail d 'Architecte sont très chronophages en phase de Specification. N'hésitez pas à leur adjoindre un ou deux collègues : ainsi, le responsable des tests pourrait utilement seconder le Designer, en particulier dans la tâche de rédaction des scénarios fonctionnels.
  - Voici une proposition de constitution d'une équipe de 6 membres :
    - M1 : MOE
    - M2 : Designer
    - M3 : Architecte
    - M4 : MOA – Plan de Tests
    - M5 : Plan Qualité – Plan de Gestion des risques
    - M6 : le maitre du temps - Organigramme des Tâches – Planning et Budget
- Le chef de projet définit également le système de valeurs que partage le groupe
  - Et c'est lui qui remet les pendules à l'heure lorsque le fonctionnement du groupe s'écarte trop de ce système de valeurs.

- Enfin, un des objectifs du Plan Management est de définir comment régler les conflits dans le groupe.
    - Vous le verrez, dans le projet DELIRE, les phases de Spécification et de Structuration se passent assez bien.
    - Par contre, en phases de Réalisation et de Convergence, c'est là que tous les ennuis convergent
    - Et quand les ennuis arrivent, c'est là que les tensions apparaissent dans le projet.
    - Le Plan Management doit donc définir comment le chef de projet travaille avec chacun des membres de l'équipe pour être sûr que tout se passe bien et que certains problèmes à venir ne vont pas dégénérer
        - Certaines personnes sont soupe au lait, mais ne gardent pas de rancune
        - D'autres sont des taiseux, mais quand ils exposent c'est très violent et souvent sans possibilité de retour arrière
        - Le chef de projet passe beaucoup de temps à comprendre comment fonctionne chacun des membres du projet.
    - Un conflit d'idée, c'est très bien si les gens se respectent et si un processus de décision ultime est défini et accepté par tous. Un conflit de personne c'est mortifère.

Rappelez-vous que le rôle du chef de projet n'est pas de prendre toute la misère du monde sur les épaules ou de pallier à la déficience de certains membres de l'équipe. Le rôle fondamental du chef de projet est de permettre à l'équipe d'experts constituée de travailler en bonne harmonie.
Pour simplifier, le Plan Management c'est le Plan de Gestion des Risques humains.

<u>Le Plan de Tests</u>

- Il est rédigé par le Responsable Tests
- Les tests sont exécutés selon quatre phases séquentielles :
- Les Unit Tests :
    - Chaque responsable de composant a la responsabilité d'écrire les Unit Tests de son composant
    - Ces Unit Tests doivent permettre au développeur de certifier qu'il a éliminé toutes les erreurs envisageables de son composant logiciel (fonctionnalité, performance, fiabilité…), de façon à ce que les tests d'intégration (le moment où on assemble tous les composants logiciels) n'aient à mettre en évidence que des problèmes d'intégration (incohérence entre un API de programmation et son appel par exemple)
    - La responsabilité du Responsable Tests est donc limitée à la spécification d'un template d'écriture de Unit Test
    - Les Unit Test sont initialement exécutés durant la phase de Réalisation.
    - Ils seront ensuite systématiquement rejoués lors d'une intervention (maintenance ou évolution) sur le composant logiciel
    - Dans ce cadre, ils doivent être automatiques.
- Les Integration Tests :
    - Les Integration Tests doivent permettre la cohérence des différents composants du produit final, par exemple la cohérence entre un API de programmation et son appel par un autre composant.

    - Les Integration Tests sont exécutés en tout début de la phase de Convergence
    - Ils seront ensuite systématiquement rejoués lors d'une intervention (maintenance ou évolution) sur un des composants logiciels du produit final, après sa relivraison.
    - Dans ce cadre, ils doivent être automatiques.
- Les Function Tests :
    - Les Function Tests doivent permettre de valider la cohérence du produit final vis-à-vis des Spécification Fonctionnelles Générales.
    - Les Function Tests sont exécutés à partir de scénarios fonctionnels, en particulier ceux définis par le Designer en phase de Specification
    - Les Function Tests sont exécutés une fois les Integration Tests convergés.
    - Les Function Tests impactés (et eux seuls en général) seront ensuite rejoués lors d'une intervention (maintenance ou évolution) sur un des composants logiciels du produit final, après sa relivraison et la réexécution des Integration Tests.
    - Les Function Tests présupposent des interactions utilisateur, ils sont donc a priori manuels, mais peuvent être avec profit automatisés via des outils de Record / Replay. Avec le risque de devoir réenregistrer le test si l'interface utilisateur change.
    - C'est durant les Function Tests qu'on validera les objectifs Qualité (performance, convivialité, fiabilité…) du produit définis dans le Plan Qualité
- Les System Tests
    - Ils ont pour objectif de tester le produit en fonctionnement réel, y compris erratique
    - Ils doivent valider en particulier le comportement acceptable du produit en cas de mauvaise utilisation du produit par l'utilisateur
    - Ils doivent permettre de valider les messages d'erreur et leur compréhensibilité
    - Ils doivent permettre de tester le produit à ses limites (Stress Tests) : dépassement du nombre maximum d'utilisateur, chute du serveur, requête chargeant une partie monstrueusement importante de la base de données, undo intempestif….
    - Certains System Tests peuvent être automatisés, d'autres sont manuels, d'autres enfin ne sont pas documentés mais demande un certain temps de tests (cas de l'utilisation erratique du produit final)

Remarque

- Lors de la remise du produit au client, des tests de recette seront exécutés sous la responsabilité du MOA, pour valider l'adéquation du produit à ses spécifications.
- Ce sont donc des tests de type fonctionnels.
- Le MOA et le MOE se mettent d'accord en début de projet, une fois les Spécifications Fonctionnelles Générales avalisées, sur la définition exacte des tests de recette.
- A l'évidence, les tests de recette doivent faire partie intégrante de la définition des Function Tests

Les Spécifications Techniques Détaillées

- Chaque Spécification Technique Détaillée de composant logiciel est rédigée par le développeur responsable dudit composant.
- Les Spécifications Techniques Détaillées définissent les interfaces accessibles par d'autres composants (APIs de programmation)
- Les Spécifications Techniques Détaillées des composants logiciels de présentation définissent la liste exhaustive des commandes (requêtes http) accessibles
- Les Spécifications Techniques Détaillées définissent la liste exhaustive des composants logiciels sur lesquels elles ont visibilité : en d'autres termes la liste des APIs de programmation qu'elles peuvent appeler.
- Les Spécifications Techniques Détaillées définissent tous les internes du composant : structures de données, algorithmes, gestion des erreurs…
- Les Spécifications Techniques Détaillées définissent les Unit Tests de validation du composant
    - Unit Tests boite noire : on teste les APIs de programmation
    - Unit Tests boite blanche, on teste le fonctionnement et la logique internes du composant
    - Pour mémoire, un Unit Test se fait, avant le processus d'intégration, en simulant les autres composants
- Les Spécifications Techniques Détaillées rappellent les objectifs qualité (en particulier performance) à atteindre, et la méthode pour le mesurer.
- Bien entendu, les Spécifications Techniques Détaillées présupposent le respect des normes de programmation définies dans le Plan Qualité.

L'objectif est qu'une Specification Technique Détaillée soit auto-portante. En d'autres termes, qu'elle puisse être confiée à une personne qui ne fait pas partie du projet mais qui a le savoir-faire technique en termes de développement, et que cette personne puisse faire le travail de développement d'un composant logiciel en s'appuyant uniquement sur sa Specification Technique Détaillée

Les livrables de la phase de Structuration étape 2

Les buts de la phase de Structuration étape 2 sont :

- D'avoir attribué chaque tâche à accomplir en phases de Réalisation et de Convergence à l'un des acteurs du projet
- D'avoir défini le planning et le budget des phases de Réalisation et de Convergence

Avec le Plan Qualité, le Plan Management, le Plan de Gestion des Risques, le Plan de Tests et les Spécifications Techniques Détaillées, on commence à avoir une bonne idée des tâches à réaliser en phases de Réalisation et de Convergence. Pour être tout à fait précis, ce qui manque pour être exhaustif c'est la rédaction des livrables finaux, la rédaction des documentations, la rédaction de la conclusion générale et des post mortem, et la rédaction et la préparation de la soutenance finale et de la démonstration finale

L'Organigramme des Tâches

- Il est rédigé par le responsable Planning
- Il liste
    - L'ensemble des tâches à réaliser
    - Et pour chacune des tâches

- La personne qui en a la responsabilité
- Les tâches prérequisites
- Le coût (en nombre d'heures) de la tâche

Attention un Organigramme des Tâches peut comporter plusieurs centaines de tâches

Le Planning

- Il est rédigé par le responsable Planning
- Dans cette seconde version, on va tenter de définir le planning et le budget des phases de Réalisation et de Convergence
- La source d'information est l'Organigramme des Tâches. Pour chacune des tâches on connait :
  - La personne qui en a la responsabilité
  - Les tâches prérequisites
  - Le coût (en nombre d'heures) de la tâche
- C'est donc la mission du responsable de Planning de trouver une façon pertinente de positionner ces différentes tâches sir le calendrier, en faisant démarrer la phase de Réalisation juste après les vacances de Noël.
- Le planning est considéré comme correct quand
  - Une tâche ne démarre pas avant la fin de ses tâches prérequisites
  - Un personne ne fait pas deux tâches simultanément
  - Une personne ne dépasse pas un nombre d'heures raisonnable au cours d'une même semaine
  - La charge de travail est harmonieusement répartie entre les membres du groupe
  - Et le projet termine en temps et en heures.
- Sachez vous garder un buffer. Je répète régulièrement depuis plus de 20 ans que dans un projet il manque toujours 15 jours à la fin.

Le Budget

- Il est construit par le responsable Planning
- Ici nous ne parlons pas d'€ mais d'heures de travail.
- Le Budget est obtenu en traçant pour chaque semaine la courbe d'heures de travail effectuées.
  - L'objectif du budget est d'avoir une prévision pour pouvoir comparer la réalité avec cette prévision durant les phases de Réalisation et de Convergence
  - Il y a un budget pour l'équipe, et un budget par membre de l'équipe :
  - L'un comme l'autre doivent être régulier dans le temps.
  - En toute logique, la période où on travaille le plus est la phase de Réalisation : chacun travaille dans son coin en n'ayant aucune dépendance vis-à-vis des autres membres de l'équipe.
- On présente chaque budget sous forme de 2 courbes : une courbe instantanée, une courbe cumulée.
- N'oubliez pas d'inclure dans vos budgets les heures de réunion et les heures de management du chef de projet.
- Attention 1 heures de réunion où les 6 membres sont présents coûte 6 heures dans le budget du projet.

Les livrables de la phase de Réalisation

Le but de la phase de Réalisation est de convertir les Spécifications Techniques Détaillées, issues de la phase de Structuration et qui sont rédigées en langage humain, en code qui est un langage compréhensible par une machine. Le tout en utilisant des patterns qui maximisent la qualité (fiabilité, performance, convivialité…) du produit final.
La phase de Réalisation n'est absolument pas une période de créativité, mais une période d'expertise technique
En toute logique, si les Spécifications Techniques Détaillées ont été bien rédigées, cette phase doit être très rapide. Du moins elle devrait l'être dans un monde idéal. Hélas, un projet informatique ressemble rarement à un dessin animé de Walt Disney.

Y a-t-il des livrables en phase de Réalisation ? La réponse qui vient spontanément à l'esprit serait « NON, on ne fait que du coding ». C'est en fait un peu restreint comme réponse

Votre code est destiné à perdurer, et doit être considéré comme un livrable, avec des critères de qualité.

- Vous serez amenés à le faire évoluer dans le cadre de la version 2 et de la version 3 qui vont suivre dans quelques mois ou quelques années. Et vous serez surpris de voir à quelle vitesse on oublie pourquoi on a fait tel ou tel choix technique ; surtout s'il s'agit d'une astuce. .
- Dans un an, voire dans 10, quelqu'un d'autre que vous devra faire en urgence la maintenance de votre composant logiciel, suite à un incident majeur chez un client clé : il est donc important que votre code soit bien documenté en interne (c'est-à-dire au travers de commentaires pertinents) pour permettre d'intervenir rapidement est avec sérénité.

Documenter le composant logiciel dans un document externe n'est jamais une bonne solution

- D'une part le document n'est rapidement plus à jour
- D'autre part personne ne sait pas trouver ledit document le jour de la crise de maintenance

Durant les phases de réalisation et de Convergence chaque développeur devra

- D'une part loguer chaque semaine son activité : nombre de lignes de code pissées, nombre d'erreurs trouvées, nombre d'erreurs corrigées…,
- D'autre part, chaque livraison de composant logiciel sera accompagnée d'une fiche de livraison, selon un template défini par le responsable du Plan Qualité, qui résume les points clé du travail effectué : nombre d'heures de travail, difficultés rencontrée, fonctionnalités livrées, unit tests effectués, validation des performances attendues, nombre d'erreur trouvées, nombre d'erreurs corrigées…

Ces suivis d'activité et l'ensemble des fiches de livraison vous permettront de refaire en fin de projet DELIRE l'historique des phase de Réalisation et de Maintenance pour en tirer des ordre de grandeur qui nourriront votre expertise, et des enseignements sur des erreurs à ne pas re-commettre.

Les livrables de la phase de Convergence
En fin de phases de Spécification puis de Structuration, vous avez mis au point des livrables : Specification Fonctionnelles Générales, Architecture Logique, Plan Qualité, Plan Management, Organigramme des Tâches, Plan de Tests…

En fin de phases de Réalisation puis de Convergence, la réalité sera plus ou moins proche de ce qui avait été initialement prévu. Et l'honnêteté me conduit à vous dire que conception et réalisation seront souvent assez divergentes.
Il y a toujours d'excellentes raisons pour ces divergences :

- Architecture Logique peu robuste
- Conception trop ambitieuse
- Difficulté à reprendre le projet en janvier
- Difficulté technique non identifiée…

Il n'y a pas de jugement de valeur dans ces difficultés rencontrées. Au contraire, elles font partie de l'enseignement de DELIRE… si on sait les expliquer et si on est capable de proposer une démarche pour ne plus tomber dans les mêmes pièges lors de votre prochain projet (je vous rassure, vous tomberez sur d'autres pièges)

Chaque responsable de livrable de Spécification ou de Structuration devra reprendre son livrable et identifier :

- Ce qui était prévu et qui a été réalisée 
- Ce qui était prévu et qui n'a pas été réalisé 
- Ce qui n'était pas prévu et qui a été réalisé 

Mais il est également nécessaire d'expliquer les raisons qui ont conduit à ces renoncements et à ces oublis.
Et ensuite il faut en tirer des enseignements. Ce qui est important dans DELIRE n'est finalement pas ce que vous réalisez mais les enseignements que vous tirez des difficultés auxquelles vous aurez été confrontés.

Certaines documentations seront nécessaires pour déployer le produit en environnement industriel.

- Documentation Utilisateur (si possible en ligne)
  - La finalité du produit
  - Les principes de base d'utilisation du produit : le menu, l'undo, les raccourcis, la liste des commandes permanentes, la liste des commandes en fonction du profil de l'utilisateur…
  - La liste des messages d'erreur susceptibles d'apparaitre
  - Et le comportement à adopter en cas de problème et/ou de message d'erreur.

  A priori, si l'I.H.M. a été bien pensé, la documentation stricte de chaque interaction pour chaque commande n'est pas une nécessité.
- Documentation d'installation
  - L'essentiel dans DELIRE est l'installation du serveur : sur le site du client ou via un IaaS dans un Datacenter type AWS
  - Dans les deux cas, il est nécessaire de dimensionner la ou les machines, de générer l'adresse URL du site, et de créer à minima le compte de l'administrateur.
  - Le plus simple serait de lancer une procédure qui poserait des questions et qui générerait automagiquement l'infrastructure idoine.
- Documentation d'administration.
  Le travail de l'administrateur de site a pour objectif de
  - Générer la structure arborescente de l'hôpital

    - Créer les nouveaux utilisateurs (médecins, infirmières, secrétaires…) incluant son savoir-faire (neurologue, cardiologue…), d'obtenir un password en réponse et de le transmettre au nouvel utilisateur
    - Rattacher chaque nouvel utilisateur à un nœud de l'arborescence de l'hôpital
- Documentation de maintenance
    - Si vous n'expliquez pas très exactement quelle est la démarche à suivre pour soumettre un incident à l'équipe de maintenance, vous risquez de vous retrouver submergés par les appels téléphoniques et les cris d'orfraie.
    - Dans la documentation de maintenance
        - L'importance du problème pour le client
        - La langue qu'il faut utiliser (pour avoir reçu un incident rédigé en chinois, je peux vous dire que c'est surprenant)
        - L'identification du rôle, de la commande
        - La description du problème
        - Si besoin, les données nécessaires pour pouvoir reproduire le problème

  Pour information, la maintenance n'est pas un service gratuit. Le standard est de le facturer 15% du prix d'achat… chaque année.

Enfin il faut conclure

- L'équipe rédige conjointement la conclusion finale, qui revient sur le projet et en tire des enseignements
    - Ce qui a bien marché et pourquoi
    - Ce qui n'a pas bien marché, pourquoi, et les enseignements tirés pour vos prochains projets
    - Ce qui peut être amélioré pour les projets DELIRE des années suivantes
- Chacun rédige personnellement son post mortem
    - Ce qu'il a apprécié et pourquoi
    - Ce qui n'a pas bien marché, pourquoi, et ce qu'il peut faire pour mieux gérer les projets dans lesquels il sera impliqué dans le futur
    - Sans que cela soit une nécessité, les conseils qu'il peut donner aux autres membres de l'équipe pour mieux gérer les projets dans lesquels ils seront impliqués dans le futur

La conclusion générale et les post mortem ne sont pas des règlements de compte. Un projet se fait et se gagne en équipe. Et l'essentiel est de trouver des voies pour s'améliorer personnellement

Ha, petit détail avant de considérer le projet DELIRE comme terminé, il reste la présentation et la démonstration à rédiger….
Après, vous vous sentirez délivrés. C'est peut-être pour cela qu'en anglais les livrables s'appellent « deliverables ».

Le projet DELIRE c'est