SI & BDD

Démarche projet

MOA: Maître d'ouvrage
Personne qui exprime le besoin, les objectifs, les délais, le budget. Décide du lancement du projet et confie la réalisation à la MOE
MOE: Maître d'ouvre
S'engage à assurer que la solution obtenue est conforme aux spécifications fonctionnelles signées par la MOA. Doit concevoir et mettre en oeuvre les livrables attendues par la MOA
AMOA: Assistance à la maîtrise d'ouvrage
Aide le client à identifier ses besoins et les formuler. Une fois les besoins établis, aide à trouver la meilleure solution en fonction des contraintes
Recueillir et analyser le besoin client: interviews individuelles, réunions, questionnaires. Tout ça dans l'expression de besoins
Types de gestions de projet, schémas: document annexe

Scrum

Définition agilité:
Méthode de gestion de projet itérative centrée sur la souplesse et la performance. Centré sur l'humain et la communication, fait participer les clients au dévellopement du produit tout au long de l'avancement du projet. Avantages: importance du retour d'information, très forte flexibilité, implication de l'équipe
Définition Scrum:
Framework de gestion de projet Agile basé sur la redéfinition des priorités et les cycles de livraison courts par le biais de sprints.
Acteurs:

- Product Owner (PO): la personne qui connaît le produit, ses exigences, celles des clients et du marché. Crée et gère le backlog produit, s'assure que tout le monde comprend les tâches qu'il contient, fourni les orientations pour les prochaines fonctionnalités à livrer
- Scrum Master: Gère le fonctionnement Scrum, organise le daily meeting, la planification de sprint, la revue et la retrospective du sprint
- Team: Idéalement aux compétences variées, s'organise de façon autonome et collective pour développer le produit. Prévoit la quantité de travail

Sprint:
Délai réel dont dispose l'équipe pour terminer un increment. Plus le travail est complexe, plus les sprints doivent être courts idéalement. Commence par une planification du sprint qui détermine l'objectif du sprint. Les users stories sont alors ajoutées au sprint backlog. A la fin de cette planification chacun doit savoir ce qu'il a à faire. Un sprint fini par une revue de sprint, réunion informelle de démmo ou d'inspection de l'incrément. Il y a alors livraison ou non. Cette revue peut avoir une incidence sur le prochain sprint par le biais d'une modification du backlog produit
Stand-up meeting/daily Scrum:
Réunion matinale quotidienne, même heure, même endroit, debouts. Doit être rapide, permet de vérifier que l'équipe est en phase avec l'objectif du sprint, s'organiser pour les 24 heures à venir,

témoigner de ce qui a été fait, de ce qui est à faire et d'éventuelles problématiques
Retrospective de sprint:
Documentation de ce qui a fonctionné ou pas (sprint, projet, outils...) en ce concentrant sur ce qui a bien fonctionné et les éventuelles améliorations plutôt que sur les échecs. REX (retour d'expérience)
Increment/Itération:
Livrable obtenu pendant le sprint
Backlog produit:
Liste principale des tâches (user stories) à réaliser qui fait office de point de départ pour le backlog de sprint. Géré par le PO. Liste de fonctionnalités, exigences, améliorations... Régulièrement repensé, priorités redéfinies
Backlog sprint:
Liste d'éléments, de user stories ou de correctifs sélectionnés lors de la planification. Elements issus du backlog produit. Flexible et peut évoluer pendant le sprint, mais l'objectif final du sprint ne peut pas être remis en question
Expression de besoins:
En amont du projet, document synthétique et clair décrivant simplement le besoin sans entrer dans la solution à apporter. Par contre, le cahier des charges est un document complet qui détaille précisément le fonctionnement du projet fini
User stories:
Issues de l'expression de besoins les users stories définissent des fonctionnalités du point de vue de l'utilisateur final (en tant que, je veux, afin de). Déterminées et priorisées elles constituent le backlog produit
Definition Of Ready:
Critéres permettant de déterminer si une user story est prête à être traitée durant un sprint (relue par le PO, par un membre de l'équipe, estimée par l'équipe, validée par le métier)
Definition Of Done:
Critéres permettant de déterminer si une user story est considérée comme terminée (testée, revue de code, documentée)
Différences Kanban/Scrum: en Kanban les rôles sont flexibles, pas d'engagement sur un nombre de user stories (une par une), pas de restriction de temps, possibilité d'ajouter des user stories dans le tableau, tableau Kanban peut être partagé par plusieurs équipes produit
Différences XP/Scrum: XP met l'accent sur l'automatisation des tests, la livraison rapide de logiciels fonctionnels, équipes auto-organisées, outils de collaboration

Merise

Merise: méthode d'analyse, de conception et de gestion de projet informatique. On passe par des étapes de modélisation du système informatique de plus en plus concrètes
Dictionnaire de données: Etape de listing des données à modéliser. Chacune des données fait l'objet d'une ligne dans un tableau qui comporte ces colonnes:

- Nom court ou nom codé (code mnémonique): nom du champ final dans la BDD. Ex: les 3 premiers caractères de l'entité, tiret bas, nom de la propriété
- Nom complet ou désignation
- Type ou format: entier, flottant, champ alphanumérique, texte, date, booléen. Lettre plus légende (E pour entier, A pour alphanumérique...)

- Longueur ou taille: nombre de caractères. 1 pour un booléen. Ces deux colonnes peuvent être combinées
- Commentaire : toutes les explications supplémentaires nécessaires se trouvent ici. Parfois on trouve aussi une colonne « règle de calcul » distincte

Modèle conceptuel de données: étape suivante. Le MCD prend en compte les règles de gestion, mais pas les contraintes technologiques de réalisation de base de données. On commence par regrouper les datas du dictionnaire de données en des tout cohérents afin de créer des entités (nom de l'user, âge de l'user...) et en déterminant des attributs. Un identifiant doit être choisi, au besoin ajouter un id. L'identifiant est souligné et placé en tête de liste. On matérialise les relations entre entitées grâce à un verbe qui doit idéalement être compréhensible dans les deux sens. Noter les cardinalités (0, 1, N)
Les cardinalités N,N feront l'objet de tables d'association qui peuvent contenir des attributs (pas obligé). Une ligne de table d'association est identifié grâce à un couple d'identifiants
Justifier les choix de conception du MCD lui fera plaisir...

Dépendances fonctionnelles:
Fonctionnelles: A et B, on dit que B dépend fonctionnellement de A (A -> B) si à chaque valeur de A correspond une seule valeur de B (numéro sécu -> nom famille mais pas l'inverse)
Elementaires: A et B, A -> B est élémentaire si il n'y a pas de C contenu dans A tel que C -> B (n°commande, n°article -> nom_article n'est pas élémentaire, n°article -> nom_article l'est)
Directes: A et B, A -> B est directe si il n'y a pas de C tel que A -> C puis C -> B
Première forme normale (tous les attributs de la relation autres que la clef primaire ont une dépendance fonctionnelle avec la clef primaire)
Deuxième forme normale (la relation est en première forme normale et les dépendances fonctionnelles sont élémentaires)
Troisième forme normale (la relation est en deuxième forme normale et les dépendances fonctionnelles sont directes)
Bien sûr on vise la troisième forme normale

Modèle logique de données: étape suivante. Représentation qui prend en compte le choix technologique de réalisation de la BDD. On supprime les cardinalités et on intègre dans les entités les clefs étrangères correspondantes. Les tables d'association deviennent des entités à part entière. Les clefs étrangères arrivent dans l'entité du côté du "1" dans une relation "1,N"

En cas d'intégrité fonctionnelle il faut une cascade à la création ou à la suppression

Exemples: annexe

UML/Diagramme cas d'utilisation

UML: Langage de modélisation unifié (Unified Modeling Language) méthode normalisée de visualisation à base de pictogrammes pour faciliter la rédaction de documents nécéssaires à la réalisation d'un projet informatique. Permet la description d'un système d'une manière compréhensible pour les intervenants externes non développeurs

Diagrammes de cas d'utilisation:
On représente le système en lui-même en tracant un rectangle qui permet de figurer ce qui fait partie du système et ce qui gravite autour. On donne un nom à ce système en le nommant au-dessus du rectangle. Ce qui se trouve dans le rectangle doit être informatisé
On commence par représenter les fonctionnalités qui représentent des cas d'utilisation. Une

fonctionnalité est une forme ovale dans le rectangle contenant un verbe ou une phrase
Ensuite on positionne des acteurs, bonhommes baton, à l'extérieur du rectangle. On les place à gauche pour les acteurs primaires, ceux pour qui l'application à été crée, à droite pour les acteurs secondaires, les internes au système (ex: administrateur)
On continue en associant des acteurs et des fonctionnalités pour savoir qui fait quoi (flèches)
Les acteurs peuvent hériter entre eux. La flêche désigne la nature du profil (-> "est une sorte de")
Les fonctionnalités peuvent aussi hériter comme les acteurs. On représente une contrainte grâce à une flèche en pointillées portant la mention "includes". La flèche pointe vers la fonctionnalitée requise. On peut représenter une fonctionnalitée comme étant une extension d'une autre fonctionnalitée également grâce à une flèche en pointillées portant la mention "extends". La flèche pointe de l'extension vers la fonctionnalitée qui est étendue

Exemple: annexe