h e g

Haute école de gestion
Genève

Dynamic_Assoc'

Évaluation des méthodologies – Version 1.6

Historique des révisions

Date	Version	Description	Auteur(s)
6.11.2017	1.0	Rédaction initiale	AF
10.11.2017	1.1	Ajouts DAD	SG
12.11.2017	1.2	Modification matrices	AF
12.11.2017	1.3	Mise en page, orthographe	SG
04.12.2017	1.4	Correction suite à A2, explication des matrices	AF, SG
Avril 2018	1.5	Modification suite à l'arrêt de collaboration avec le CODAP	SG
Mai 2018	1.6	Mise en page finale	SG

Forbat Adrien
Gloor Sara

Projet de GREP
Mai 2018

Table des matières

1. Choix du processus

Avant de se lancer plus en avant dans ce projet, il est primordial de savoir avec quel processus celui-ci devra être réalisé.
En effet, chaque processus à ses avantages et inconvénients et il serait préférable d'utiliser la méthode la plus adéquate au bon accomplissement des différentes tâches et attentes du mandant.

Comparaison entre les différents processus

Ci-dessous, une étude comparative entre les processus RUP, SCRUM et DAD :

Caractéristiques	RUP	SCRUM	DAD
Maintenabilité	RUP fournit une grande quantité de documentation. À chaque itération, plusieurs documents sont réalisés, ce qui permet d'avoir une bonne traçabilité du projet et un produit maintenable.	SCRUM ne fournit aucune documentation. Ceci pose énormément de problèmes pour la maintenance du logiciel, car aucun document ne permet de comprendre ni son fonctionnement ni son architecture.	DAD est le juste milieu, il fournit le "just enough" en termes de documentation. Cela permet de rendre le logiciel maintenable tout en évitant de submerger le mandant de document, mode d'emploi et « whites papers » en toutes sortes.
Livraison d'une solution qui s'intègre à l'environnement du client	RUP est un processus long, généralement plus d'une année, mais le livrable final s'intègre parfaitement à l'écosystème du mandant.	SCRUM ne possède pas d'Architecture Owner alors que ce rôle est essentiel puisqu'il permet de vérifier que le livrable final est d'architecture semblable à celle du client et donc que le livrable final s'intégrera au système de l'entreprise. Les risques sont donc très élevés en ce qui concerne l'intégration.	DAD propose **une solution** au problème de l'entreprise et cette solution s'adaptera et s'intégrera à l'écosystème du mandant. C'est pourquoi dans DAD on parle de solution répondant à un problème métier et non plus simplement de « logiciel ».
Collaboration avec le mandant	À chaque demande du mandant, un document est créé pour l'équipe, ce qui laisse peu d'interaction personnelle.	Un Product Owner est présent du côté client afin de faire connaître les différentes spécifications ainsi que la priorité de celles-ci tout au long du projet.	La collaboration avec le mandant est très importante et mise en avant avec DAD.
Analyse profonde des spécifications	RUP dispose d'une phase d'inception qui va permettre de fortement documenter l'analyse des spécifications afin de réellement les comprendre et de rendre un logiciel qui correspond aux spécifications émises par le mandant.	SCRUM c'est : se mettre à coder très rapidement, sans pour autant avoir forcément bien compris et analyser les spécifications ou besoins du mandant. Il n'y a pas de séparations en phases. Les développeurs se lancent dans le développement le plus rapidement possible.	DAD dispose également d'une phase d'inception durant laquelle seront rigoureusement analysées les spécifications du client. Un document de vision ainsi qu'une liste des risques seront établis afin de comprendre au mieux les attentes du mandant. Lorsque tout sera évalué et accepté par le mandant (on évite donc les incompréhensions), le développement du logiciel pourra commencer.
Risk-value lifecycle	Tous les risques possibles vont être analysés afin d'éviter les problématiques de retards de livraison ainsi que de projets non aboutis.	Aucune étude des risques n'est faite dans SCRUM, ce qui signifie que tout au long du projet, celui-ci peut potentiellement connaître de gros retards de livraison et dans le pire des cas, le logiciel pourra même être impossible à livrer.	Durant la phase d'inception (phase initiale du projet), tous les risques possibles vont être étudiés, analysés et documentés. Tout comme dans RUP.

Caractéristiques	RUP	SCRUM	DAD
Réactivé au changement	La réactivité au changement est quasi inexistante, puisqu'il faut attendre la fin d'une itération, déjà très longue, afin d'évaluer si cela est faisable ou non.	Les changements dans les spécifications peuvent être traités durant tout le projet.	Les changements dans les spécifications peuvent être traités durant tout le projet.
Livraisons fréquentes	Les itérations sont très longues donc les livraisons peu fréquentes. Lorsqu'une modification doit avoir lieu il faut obligatoirement attendre que l'itération en cours soit terminée.	Les itérations sont courtes (2 à 4 semaines) afin de fournir des livrables le plus rapidement possible. Ces itérations sont adaptables donc il est facile de faire des modifications en cours de route.	À l'instar de SCRUM, DAD fournit des itérations courtes qui permettent de fournir des livrables rapidement en fonctionnant de manière itérative et incrémentale. Cela permet d'effectuer des modifications suite à des spécifications mal comprises.
Meetings fréquents ente les membres du team	La méthode ne spécifie rien au sujet des meetings.	Tous les jours ont lieu des Daily Meetings.	Tous les jours ont lieu des Daily Meetings.
Définition des rôles	Avec RUP, chaque développeur a son rôle clairement défini pour chaque tâche. Cela permet une certaine stabilité, mais ne fournit donc pas de polyvalence.	Il y a un Product Owner ainsi qu'un Scrum Master ayant leurs rôles à jouer, mais pour tout le reste du team (il n'y a pas de chef de projet) c'est une gestion sous forme d'auto-organisation. Aucun membre du team n'a de rôle clairement défini et il peut changer constamment.	Il y a un Product Owner, un Team Lead ainsi qu'un Architecture Owner. Chacun a son rôle bien défini. Cependant, pour les membres du team (développeurs) ils n'ont aucun rôle de défini.

Matrice préférentielle[1]

Afin d'évaluer quelle méthodologie sera la meilleure à utiliser pour notre projet nous avons réalisé une matrice préférentielle :

No.		a	b	c	d	e	f	g	h	i
	Eléments de solution									
a	*Maintenabilité (documentation)*									
b	*Livraison d'une solution qui s'intègre à l'environnement du client*	a								
c	*Collaboration avec le mandant*	a	c							
d	*Analyse profonde des spécifications*	c	d	d						
e	*Risk-value lifycycle*	a	e	c	d					
f	*Réactivité au changement*	a	f	c	d	f				
g	*Livraisons fréquentes*	a	b	c	d	e	f			
h	*Meetings fréquents entre les membres du team*	a	h	c	d	h	f	h		
i	*Définition des rôles*	a	i	c	d	e	f	g	h	

[1] [1] Une explication des matrices préférentielle est donnée en dernière page du document (page 12)

No.	Eléments de solution	Nombre (36)	Ordre	Pondération %	Pondération modérée %
a	*Maintenabilité (documentation)*	7	1	19,44	20,00
b	*Livraison d'une solution qui s'intègre à l'environnement du client*	1	9	2,77	2,00
c	*Collaboration avec le mandant*	7	3	19,44	16,00
d	*Analyse profonde des spécifications*	7	2	19,44	17,00
e	*Risk-value lifycycle*	3	6	8,33	10,00
f	*Réactivité au changement*	5	4	13,88	13,00
g	*Livraisons fréquentes*	1	7	2,77	5,00
h	*Meetings fréquents entre les membres du team*	4	5	11,11	12,00
i	*Définition des rôles*	1	8	2,77	5,00

Nous comparons chaque numéro les uns avec les autres afin de déterminer lesquels sont les plus importants dans notre situation.

Grâce à cela, nous pouvons déterminer la pondération d'un point et ensuite réaliser une matrice multicritère qui nous indiquera quel processus de gestion de projets est le plus adapté à nos besoins.

À noter que pour le point concernant une solution qui s'intègre à l'environnement du client, nous lui avons donné très peu d'importance sachant que notre solution ne viendra pas en intégration, mais en module séparé et distinct de celui déjà en place.

Matrice multicritère

Grâce à ces matrices nous pouvons évaluer les différentes méthodologies selon leurs critères, avec les pondérations définies précédemment.

Cette évaluation passe par une analyse dite multicritères :

Critères (obligatoires)		RUP		SCRUM		DAD	
Maintenabilité (documentation)		***oui***		***non***		***oui***	
Critères (facultatifs)	Pondération	Note	Note pondérée	Note	Note pondérée	Note	Note pondérée
Livraison d'une solution qui s'intègre à l'environnement du client	**1**	*8*	8	*2*	2	*10*	10
Collaboration avec le mandant	**7**	*4*	28	*10*	70	*10*	70
Analyse profonde des spécifications	**7**	*10*	70	*5*	35	*10*	70
Risk-value lifycycle	**3**	*10*	30	*1*	3	*10*	30
Réactivité au changement	**5**	*3*	15	*10*	50	*8*	40
Livraisons fréquentes	**1**	*2*	2	*10*	10	*7*	7
Meetings fréquents entre les membres du team	**4**	*3*	12	*10*	40	*10*	40
Définition des rôles	**1**	*1*	1	*10*	10	*6*	6
Σ			**158**		**218**		**263**

Après analyse, nous pouvons facilement nous rendre compte que la méthodologie de gestion la plus adaptée pour notre projet est DAD.

DAD nous permettra de fournir rapidement un logiciel complet, répondant aux attentes du client et maintenable.

À noter toutefois que même si le processus SCRUM a obtenu un bon résultat global, il n'aurait pas été sélectionné, car il répond négativement au critère que nous avons considéré comme obligatoire. SCRUM ne fournissant aucune documentation, la maintenabilité du logiciel pour le futur est compromise.

2. DAD – Disciplined Agile Delivery

Qu'est-ce qu'est DAD ?

Le nombre d'entreprises appliquant les pratiques agiles a augmenté ces dernières années. Le succès de leurs projets n'est cependant pas toujours une réalité. Une étude de Scott Ambler (2013) a fait ressortir deux éléments qui seraient responsables de bon nombre d'échecs de ces projets menés au moyen de méthodes agiles (telles que SCRUM) :

- Le manque de discipline
- La non-couverture du cycle de vie dans son intégralité

Dans le sondage issu de la même étude, la grande majorité des entreprises "agiles" révèle tout de même faire un document de vision, une évaluation du temps et des coûts, ainsi que la modélisation de l'architecture au début du projet.
De cette tendance à appliquer des bonnes pratiques, est née la méthode agile hybride : Disciplined Agile Method (DAD). Hybride car elle combine des pratiques de méthodes agiles telles que SCRUM avec des pratiques issues de Unified Process (RUP).

Principes

- Méthode itérative et incrémentale
  - Les itérations sont courtes et de durée fixe (2 à 6 semaines).
- Méthode orientée solution
  - DAD ne s'applique pas seulement au développement logiciel, mais est également un outil pour mettre en place des solutions en tous genres face à des problèmes et projets d'envergure. On peut, par exemple, utiliser DAD pour la mise en place d'un ERP au sein d'une entreprise.
- Méthode centrée sur les personnes
  - La communication au sein de l'équipe est mise en avant, mais DAD valorise tout de même la documentation écrite pour garder une trace de ce qui se dit. La responsabilité sur le travail produit ainsi que sur la manière de le produire est partagée par toute l'équipe.

Rôles

Les principaux rôles sont :

Le Product Owner (~ côté client)

- Responsable de la vision du produit
- Responsable de la priorisation des stories (c'est lui qui impose les business values)
- Son rôle augmente avec le déroulement du projet

Le Team Lead

- Correspond au Scrum Master de la méthode SCRUM avec davantage de leadership
- Responsable de maintenir l'équipe axée sur les objectifs du projet
- Protège l'équipe des interférences externes
- S'assure que les réunions soient productives
- Encourage la communication ouverte et la collaboration au sein de l'équipe
- S'assure de la bonne application de la méthodologie DAD
- Responsable de l'amélioration du processus

L'Architecture Owner

- Mène les premiers pas dans le choix de l'architecture et la collecte des spécifications non fonctionnelles
- Conseille (n'impose pas) en matière d'architecture et de design technique
- S'assure que la solution s'intègre à l'architecture globale de l'entreprise
- S'assure que la solution se développe de façon conforme par rapport aux spécifications non fonctionnelles (surtout pour la maintenabilité)
- S'assure que la solution soit régulièrement testée et intégrée
- Peut participer au code de la solution

Les Team Members

- Collectivement responsables du code

Les rôles secondaires sont :

Les Stakeholders

- Tout individu ayant un intérêt dans le projet

Rôles annexes

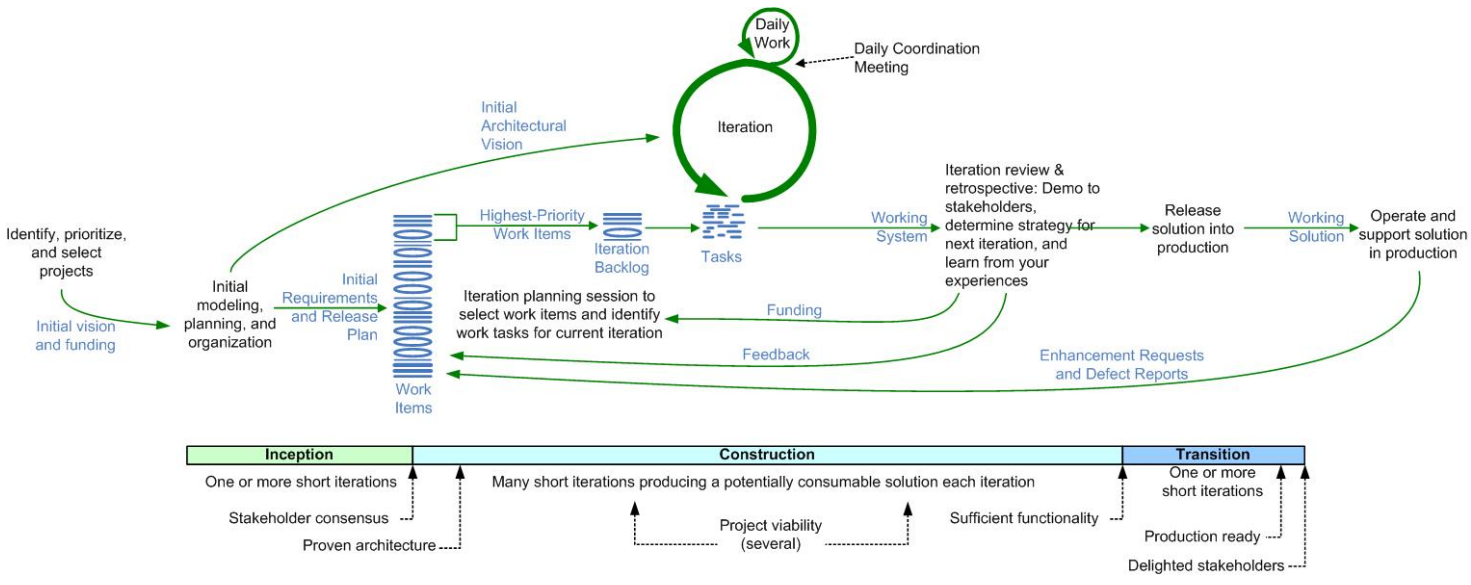
- Individus temporairement impliqués dans le projet (spécialiste métier, testeur indépendant…etc.)

Configuration d'une équipe DAD de petite taille:

- 1 Product Owner + 1 Team Lead + 1 Architecture Owner + Team Members
  Note : les rôles de Team Lead et Architecture Owner peuvent être assumés par la même personne

Cycle de vie

Il y a trois phases au sein de la méthode DAD. L'Inception, la Construction et la Transition.
DAD est une méthode itérative tout au long de ces phases.



Source : www.disciplinedagiledelivery.com

Phase d'Inception

C'est la phase de vision et de planification, avant de commencer à construire la solution.
Cette phase a pour objectif d'analyser le projet et de déterminer sa viabilité.

Voici les différentes étapes qui constituent la phase d'Inception :

- Formation de l'équipe et définition des rôles de chacun
- Compréhension du problème métier
- Identification de ce qui est réutilisable dans l'entreprise pour la solution (software engineering)
- Réalisation de la vision du projet avec le Product Owner, comprenant :
    - Le document de vision (but, champ d'application, besoins, etc.)
    - L'architecture globale (idée générale du système, choix de la technologie, intégration à la stratégie actuelle)
    - Les spécifications et leur modélisation (Work Items List, Use-cases…etc.)
    - Les spécifications non fonctionnelles
    - La planification (délais, planning d'itération, diagramme de Gantt)
    - La liste des risques
- Installation de l'environnement de travail
- Planification des milestones d'évaluation de la viabilité du projet
    - Prévoir au minimum deux milestones pendant la phase de construction
- Formation pour les outils de développement si nécessaire

L'outil DAD qui va permettre d'évaluer la charge de travail de l'entier du projet, et donc d'estimer le temps et les coûts de celui-ci, est le Planning Poker. On définit des points (évaluation de la charge de travail en jour/homme) pour chaque Work Item.

La Work Items List contient tous les Work Items, donc les uses-cases du projet. Les différents éléments y sont organisés par ordre de priorité décroissante, priorité qui est définie en se basant sur la Business Value (attribuée par le Product Owner) et le risque (attribué par le Product Owner et les membres de l'équipe).

- Chaque nouveau Work Item est priorisé et ajouté à la Work Items List.
- Les Work Items peuvent être repriorisés à n'importe quel moment.
    - C'est une des valeurs agiles de DAD
- Les Work Items peuvent être supprimés du stack à n'importe quel moment.

Une fois tout ceci réalisé, a lieu le « Stakeholder Consensus » où il sera décidé, d'un commun accord, si le projet rentre en phase de construction. C'est le "Go/No-Go".
Si les parties prenantes au projet ne parviennent pas à un accord à la fin de la phase d'inception ou si le projet est considéré trop risqué, on s'arrête là.

Phase de Construction

Cette phase est structurée par une série d'incrémentations (itérations), conformément à la méthode DAD. Chaque itération réalise une ou plusieurs spécifications, donc un ou plusieurs Work Items, et donne lieu à une partie exécutable du système à réaliser.
À la fin de chaque itération, on doit pouvoir produire une « Release » de la solution qui soit démontrable et testable auprès des parties prenantes (Stakeholders).
Il n'est pas permis d'ajouter des tâches à réaliser en cours d'itération. Chaque nouvel élément sera ajouté à la Work Items List selon sa priorité lors de la prochaine planification d'itération.

Planning d'itération :

1. Planification et vérification de la disponibilité des membres de l'équipe.

2. Sélection d'un Work Item en tenant compte de la Business Value et le facteur risque de celui-ci.

3. Obtenir les « Just-In-time Requirements » de la part du Product Owner et définir les critères qui déterminent qu'un élément est considéré comme fait.

4. Modélisation de la solution potentielle (Model Storming).

5. Décomposition du Work Item en tâches de 1 à 5 heure(s) de durée (exemple de tâches : identifier les règles métier, écrire code, écrire tests, design de l'interface utilisateur, etc.).

6. Attribuer les tâches aux Team Members. Soit on attribue toutes les tâches pendant la planification d'itération, soit chacun choisit une nouvelle tâche à chaque fois qu'il en termine une. (Bonne pratique : utiliser un TaskBoard pour l'assignation et le suivit de l'état des tâches.)

7. Mettre à jour la charge de travail sur les Work Items similaires à celui effectué lors de l'itération en cours.

8. Vérifier la santé de notre planification au moyen de la vélocité de l'équipe et du Burndown Chart. S'assurer que l'équipe n'est pas surchargée et que toutes les ressources sont utilisées.

9. Obtenir l'engagement de l'équipe sur le travail à effectuer pour une itération. Si l'engagement n'est pas tenu, il faut chercher la source du problème (incertitudes sur la charge de travail de certains éléments, mauvaise dynamique d'équipe...etc.).

Les points 2 à 7 se répètent itérativement pour chaque Work Item ajouté à l'itération. On passe au point 8 lorsque l'on a planifié assez de Work Items pour l'itération à venir (couvrant le 80% de la disponibilité des Team Members pour cette itération).

Quotidiennement :

1. Coordonner (10-15 min)
    - Meeting quotidien
    - Mise à jour du tableau des tâches
    - Mise à jour de l'itération

2. Collaborer (5-6 h)
    - Résoudre les problèmes
    - Création de tests
    - Développement de code
    - Model Storming
    - Déploiement sur l'environnement test

3. Conclure (idéalement effectué sans problème)
    - Stabiliser

Après une ou deux itérations, l'architecture choisie doit être vérifiée afin d'être sûr qu'elle couvre les spécifications non fonctionnelles.

À la fin d'une itération, on vérifie avec le Product Owner le résultat obtenu. Si le PO n'est pas satisfait sur une fonctionnalité (ou plusieurs), alors cette fonctionnalité sera réinsérée dans l'itération suivante pour être corrigée.

Après plusieurs itérations, on peut faire une « Release », c'est-à-dire la livraison d'un incrément de fonctionnalités au mandant.

Transition

Lorsque les fonctionnalités sont considérées suffisantes, le projet entre en phase de transition. Comme son nom l'indique, cette dernière phase est le moment où le projet est livré au client final.

1. Coordonner
    - Planification de la phase

2. Collaborer
    - Planification de la transition
    - Derniers tests et corrections de fin de cycle
    - Solution pilote, beta
    - Finalisation de la documentation
    - Communication du déploiement
    - Préparation de l'environnement de support
    - Formation des futurs utilisateurs

3. Conclure
    - Préparation à la production
    - Déploiement de la solution finale

Suite à cela débute l'utilisation de la solution par les intéressés, puis le feedback de ceux-ci.

Documentation & produits issus de chaque phase

Bien que les méthodes agiles telles que SCRUM plaident pour l'absence de documentation afin de ne produire que du livrable, le plus rapidement possible, DAD intègre certains documents, pour faciliter la maintenance du logiciel, entre autres.

"Just enough doc" c'est la documentation essentielle à avoir. C'est la documentation qui fait le lien entre les spécifications et l'implémentation du code (lien entre le domaine métier et le code).

Documents produits :
La documentation est réalisée durant et après chaque itération afin de documenter des éléments stables. Elle doit faire partie des critères de définition du statut "fait" d'un Work Item.

Produits issus :
Lors de la phase de construction, on produit énormément puisqu'on incrémente peu à peu les fonctionnalités de notre solution.

Inception :

- Document de vision (problème à résoudre, produit envisagé, résumé des caractéristiques du produit et lien avec les besoins, qualités non fonctionnelles principales, intervenants et responsabilités, besoins essentiels avec priorité, environnement technique, contraintes à la réalisation…)
- Planification et budget
- Liste des risques
- Modélisation de l'architecture globale de la solution
- Plan d'assurance qualité
- Work Items List

Construction :

- Liste des règles métiers servant de lien avec l'implémentation de la règle dans le code
- Documentation des tests
- Modélisation des composants
- Planification des itérations
- Spécifications fonctionnelles
    - Liste des tâches à accomplir pour cette spécification
- Code compilé et exécutable – Version partielle de la solution
- Jeux de tests
- StoryBoard (prototypage graphique)

Transition :

- Manuel de l'utilisateur
- Manuel technique
- Manuel d'installation
- Version complète de la solution

Il n'y a plus de chef de projet. On responsabilise les membres de l'équipe et on leur fait confiance.
Le fait de montrer au mandant ce que le logiciel peut faire lui permet de mieux s'exprimer sur ses réelles attentes. Il faut toujours faire exprimer le problème par le mandant afin que l'on puisse correctement cerner le problème métier.

Pourquoi DAD ?

Grâce à son processus itératif, il sera possible de faire des modifications à chaque itération et donc d'être très réactif.

DAD nous permet également de rendre des livrables testables et utilisables à chaque itération. Comme nous allons travailler sur un hébergement distinct de celui de l'association, nous aurons un environnement de test optimal et pourrons leur montrer les avancées directement sur celui-ci.

Comme la maintenabilité du projet est très importante, cette méthode impose certains documents qui vont grandement faciliter celle-ci et permettre aux membres de l'association de la gérer eux-mêmes.

Plusieurs meetings sont inclus dans cette méthodologie. Cela signifie que si certaines spécifications ne correspondent pas entièrement aux besoins du mandant, il pourra toujours nous réexpliquer ses attentes afin que l'on puisse effectuer les démarches nécessaires aux modifications.

Il est très important que notre équipe puisse collaborer de manière optimale. La bonne transmission des informations la bonne collaboration entre les membres d'une équipe, sont souvent précurseurs d'un travail bien fait. DAD comprend un certain nombre de meetings pour les membres de l'équipe afin que l'on puisse exprimer nos inquiétudes et les problèmes auxquels nous sommes confrontés. De cette manière l'on peut avancer sereinement et ne pas rester bloquer sur un point en particulier.

DAD fournit une solution complète qui s'intégrera dans l'entreprise et non pas seulement un logiciel.

Cette méthode a également pour vocation de produire une liste de tous les risques qui pourraient survenir durant le projet, afin d'être prêt dans n'importe quelle situation. Être prêt signifie : augmenter le taux de réussite du projet et par conséquent livrer une solution fiable et viable.

Lorsque nous listons les spécifications, nous attribuons les risques qui pourraient survenir dans le développement de celles-ci. Si la spécification a une forte valeur ajoutée, mais qu'elle est très risquée, elle sera développée en premier.

En effet, il ne sert à rien de développer la quasi-totalité du logiciel si au final, la dernière spécification est trop risquée et qu'elle empêche la réalisation de la fin du projet.

Modification du processus

Bien que cette méthode DAD ait été choisie, nous allons devoir l'adapter à notre situation. En effet, un certain nombre d'éléments ne sont pas réalisables en l'état.
Les itérations durent généralement 2 à 6 semaines pour une équipe à temps plein. Cependant, dans le cadre du GREP, nous ne pourrons développer de cette manière et allons devoir ajuster les itérations en fonction de nos disponibilités. Nous prévoyons de passer environ un jour par semaine à l'accomplissement de nos tâches (certainement bien plus en réalité). Dans ce sens, si nous suivons le principe de DAD, une itération pour notre projet devrait durer environ de 3 à 5 mois. On est dès lors très loin d'une méthode agile. Nous allons donc réaliser des sprints beaucoup plus courts et de la même manière, les Daily Meetings seront adaptés en Weekly Meetings.

Concernant les rôles définis dans DAD, nous serons tous des Team Members. Les autres rôles sont attribués dans le Plan d'assurance qualité (voir la dernière version).

Finalement, notre méthode sera continuellement adaptée afin de répondre au mieux aux attentes du mandant tout en étant réalisable au vu de nos disponibilités.

Complément suite à l'arrêt de collaboration avec le CODAP :

Ayant choisi de travailler avec la méthodologie DAD, il nous a été plus aisé de changer différents éléments de notre projet afin de rester agiles. En effet, grâce à cette méthodologie, nous avons réagi de manière rapide aux changements sans avoir à remettre en question l'entièreté du projet.

3. Annexe

Explication des matrices préférentielles

La matrice préférentielle contient des éléments de solution que nous jugeons important à prendre en compte dans le présent projet. Selon le projet, ces éléments peuvent différer.

Chose importante, aucun élément d'aspect financier ne doit apparaître dans cette matrice.
Si besoin est, le coût est étudié dans "le coût par point" afin de donner plus de finesse à l'analyse.

Pour la lecture et compréhension :

Chaque élément de solution est comparé, un à un, avec les autres éléments.
À chaque comparaison, de manière subjective, est défini quel élément nous semble le plus important/pertinent. Ce dernier est indiqué dans la case correspondante (cellule de croisement).
Une fois toutes les comparaisons effectuées :

- On indique le nombre total de comparaisons
- Le nombre fois que chaque élément apparaît (nombre de fois où l'élément est vainqueur d'une comparaison)
- Un ordre est défini en fonction du nombre d'apparitions
- Une pondération est effectuée ((100/nbTotal) * nbApparitions)
- Celle-ci peut être légèrement adaptée (pondération modérée) si le résultat nous semble trop petit ou trop grand. Mais il faut que cette modification, comme son nom l'indique, soit modérée, car sinon la matrice ne sert au final à rien.

Suite à cela, les résultats sont reportés dans la matrice dite multicritère.
Les valeurs de l'onglet « Pondération » correspondent au nombre de fois que l'élément est apparu dans la matrice préférentielle.
L'onglet « Note », contient également des valeurs subjectives, basées sur notre expérience, connaissance ou sur des recherches, comparatifs effectués, trouvés.
L'onglet « Note Pondérée » est simplement le résultat de la Pondération * Note
L'addition de ces notes pondérées nous donne un score global de chaque solution comparée.

Si l'aspect financier est important, alors il est possible, comme indiqué, de réaliser une intégration de coût (1 an, 2 an, n années). Il suffit de diviser le coût total de la solution par le nombre total de points obtenus par la solution afin d'avoir un coût par point.

En effet, une solution peut avoir un score global beaucoup plus élevé certes, mais elle peut également avoir un coût par point très élevé, ce qui la rend peut-être moins intéressante aux yeux du mandant.

Le choix sera fait par le mandant, nous ne lui apportons que des éléments l'aidant dans la prise de décision de ce choix.